EXPLORING USER INTERFACE CHALLENGES IN SUPPORTING ACTIVITY-BASED KNOWLEDGE WORK PRACTICES

A Dissertation
Presented to
The Academic Faculty

by

Stephen Voida

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science in the
School of Interactive Computing

Georgia Institute of Technology

August 2008

EXPLORING USER INTERFACE CHALLENGES IN SUPPORTING ACTIVITY-BASED KNOWLEDGE WORK PRACTICES

Approved by:

Dr. Elizabeth D. Mynatt, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Blair MacIntyre
School of Interactive Computing
Georgia Institute of Technology

Dr. W. Keith Edwards
School of Interactive Computing
Georgia Institute of Technology

Dr. Thomas P. Moran
IBM Almaden Research Center

Dr. Gregory D. Abowd
School of Interactive Computing
Georgia Institute of Technology

Date Approved: 16 May 2008

For Amy

And in memory of Grandpa George

ACKNOWLEDGEMENTS

This dissertation is a bit of a paradox. It is the first significant piece of research writing I have ever done that lists my name as the sole author. At the same time, it also represents years of thinking, designing, programming, studying, and writing, none of which I would have been able to accomplish without the continuous and overwhelming support of so, so many people.

Thank you, Beth and Blair, for entrusting me with your original vision for the Kimura project and for giving me the opportunity to take it and tinker with it and to make it something that I could call my own. Thanks, as well, to the rest of the project team who helped to make the project a terrific first research experience (even while setting a high standard for truly bizarre research meetings)—Greg, Joe, Scott, Klaus, Ron, Amanda, Umang, Nikitas, and Rahul.

Thank you, Jim, David, Joe, Mike, Quan, Elaine, Jess, Jeremy, and Lena, for being such a supportive community at every possible turn. I count myself lucky to have had such thoughtful lab mates, inspiring collaborators, and enduring friends to have spent the last nine years with.

Thank you, Keith, for inviting me to come play with your group at PARC, for continuing to be a sounding board for all of my crazy ideas long afterwards, for always being a source of bottomless enthusiasm, and for your enduring patience.

Thanks to the members of the Mac open source community for sharing your insights, ideas, and hard work, without which Giornata never would have been possible. Thanks especially to Tony, Richard, and Jonathan, for providing the momentum for making activity-based computing possible on the Mac, and to Chris for volunteering your time and expertise to get all the pieces to actually talk to one another.

Thank you to the National Science Foundation, Sun Microsystems, Ricoh Innovations, IBM, PARC, Steelcase, and the GVU Center for graciously supporting my research.

Thank you to everyone who provided invaluable feedback on the design of these research systems, especially those of you who took an incredible leap of faith and allowed me to install software of questionable stability on the computers you rely on most, and for months at a time.

Thank you to the members of my dissertation committee—Beth, Keith, Gregory, Blair, and Tom—for bringing your considerable expertise to bear on this research. I am most indebted to the five of you for helping me to recognize where my true passions in research lie and how best to articulate my contributions so that others might be able to expand upon them.

Thank you again, Beth, for convincing me to stay at Tech, for finding so many wonderful opportunities that I could be a part of, for allowing me the chance to travel the world (several times over), for making me feel like a part of your family, and, most of all, for guiding me to develop my own ideas, vision and voice.

Thank you, Mom, Dad, Rachel, Bob, and Kathy, for all of your support and love, especially when you didn't necessarily understand what it was that I was up to here, and for all those times that I had to miss the important things happening there.

And to Amy… there's not enough space to begin to say how much your being here through it all has meant to me. Fortunately, I suspect you already know. Above all, thank you believing that I could pull it off, even when I didn't.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The venerable desktop metaphor is beginning to show signs of strain in supporting modern knowledge work. Traditional desktop systems were not designed to support the sheer number of simultaneous windows, information resources, and collaborative contexts that have become commonplace in contemporary knowledge work. Even though the desktop has been slow to evolve, knowledge workers still consistently manage multiple tasks, collaborate effectively among colleagues or clients, and manipulate information most relevant to their current task by leveraging the spatial organization of their work area. The potential exists for desktop workspaces to better support these knowledge work practices by leveraging the unifying construct of *activity*. Semantically-meaningful activities, conceptualized as a collection of tools (applications, documents, and other resources) within a social and organizational context, offer an alternative orientation for the desktop experience that more closely corresponds to knowledge workers' objectives and goals.

In this research, I unpack some of the foundational assumptions of desktop interface design and propose an activity-centered model for organizing the desktop interface based on empirical observations of real-world knowledge work practice, theoretical understandings of cognition and activity, and my own experiences in developing two prototype systems for extending the desktop to support knowledge work. I formalize this analysis in a series of key challenges for the research and development of activity-based systems. In response to these challenges, I present the design and implementation of a third research prototype, the Giornata system, that emphasizes activity as a primary organizing principle in GUI-based interaction, information organization, and collaboration. I conclude with two evaluations of the system. First, I present findings from a longitudinal deployment of the system among a small group of representative knowledge workers; this deployment constitutes one of the first studies of

how activity-based systems are adopted and appropriated in a real-world context. Second, I provide an assessment of the technologies that enable and those that pose barriers to the development of activity-based computing systems.

CHAPTER 1

INTRODUCTION

The venerable desktop metaphor is beginning to show signs of strain in supporting modern knowledge work. Traditional desktop systems were not designed to support the sheer number of simultaneous windows, information resources, and collaborative contexts that have become commonplace in contemporary knowledge work. For example, the traditional desktop interface supports multitasking only at a very low-level, per-application window basis and support for collaboration is delegated to other applications and tools rather than being integrated into the desktop directly. Resource organization is becoming increasingly difficult, partially due to increases in the volume and types of information referenced in modern knowledge work and partially due to limitations inherent in the desktop user interface design based on decades-old assumptions about the ways computers are used and the kinds of data they process.

The desktop metaphor was developed over 30 years ago at Xerox PARC. The interaction techniques comprising the desktop user interface responded to the needs of knowledge workers and the capabilities of computer technology in that era, enabling, for the first time, the simultaneous display of several applications' output. These multi-window environments helped foster the multitasking practices that are now so central to modern knowledge work. The presence of a virtual desktop "surface" behind application windows also provided spatially oriented, persistent storage for icons representing files, application shortcuts, disk drives, and, eventually, the computer, itself. This lightweight, always-at-hand storage location offered a convenient alternative to navigating a hierarchy of folders for quickly storing and retrieving information.

As computers have grown more powerful and have been adopted far outside their origins in the workplace and as users' expectations about what computers are for and

what they can do have evolved, the desktop has evolved to enable new kinds of interactions. Ravasio and Tscherter (2007) summarize this evolutionary process as follows:

> At its start, the desktop metaphor was intended to simplify poorly structured but common tasks and operations practiced by office workers. Over the years, however, personal computers were introduced into areas with no relation to the traditional office work, and with even less obvious routines to be supported. The desktop was adapted accordingly in order to keep pace with these developments (Ravasio & Tscherter, 2007).

Some of the fundamental changes to the desktop over the last 30 years can be broadly classified as new ways to manage space on the screen, new ways to manage stored information, and new tools to connect to other users.

One of the first major extensions to the desktop metaphor was the development of virtual desktops, embodied in the Rooms system (Henderson & Card, 1986). Rooms was based on a study of knowledge workers' task management practices and acknowledged that users tend to focus their interactions within semantically meaningful clusters of windows. Over time, other window management strategies emerged, ranging from Apple's Exposé interaction technique[1] to 3-dimensional window managers like the Task Gallery (Robertson et al., 2000; Robertson et al., 2007) to the incorporation of information awareness "widgets" alongside regular application windows (e.g., Apple's Dashboard).

Different models for information storage have also begun to disrupt the original model derived from information management on the physical desktop, which maps individual documents to individual files in the hierarchical filing system and each of these documents to a single window. Piles (Mander, Salomon & Wong, 1992) and BumpTop (Agarwala & Balakrishnan, 2006) investigated grouping behaviors similar to those provided for windows via virtual desktops, but did so at the level of managing the iconic representations of documents and applications where they are stored. Some

[1] http://www.apple.com/macosx/features/expose/, accessed 1 February 2008

information types—most prominently, email, but also media files such as music and photos—have no corresponding iconic desktop representations and are managed in separate information "silos," stored separately from the user's collection of "traditional" documents and accessible only through a dedicated application (Bergman, Beyth-Marom & Nachmias, 2006). The migration to more web-based storage and manipulation of documents is extending this distance between the desktop metaphor and the documents being used; it is not uncommon to have a window be the *only* representation of a document local to the user's computer, as the file itself is stored within a larger web repository.

Finally, the desktop metaphor was designed primarily for supporting a single user; the intervening years have seen a substantial rise in users' reliance upon collaboration-focused applications like e-mail and instant messaging (IM) and much more pervasive use of remote servers to store all kinds of content. Most desktop interfaces provide relatively impoverished representations of these connected and collaborative resources. Attempts to create desktop-like collaboration interfaces (e.g., Roseman & Greenberg, 1996) have demonstrated the potential in integrating collaborative functionality into systems at a deeper level. However, despite their focus on desktop-like collaboration support, these tools are typically realized as stand-alone application windows and do not integrate into users' existing desktop interfaces or desktop work practices.

Even given the very slow evolution of the established desktop metaphor in these varied directions, knowledge workers still consistently manage multiple tasks, collaborate effectively among several colleagues or clients, and manipulate information most relevant to their current task by leveraging the spatial organization of their work area (Kidd, 1994; Malone, 1983). A common thread among all of these practices that can be leveraged to provide more appropriate computational support for knowledge work is the construct of an *activity*: a collection of tools (applications, documents, and other resources) within a social and organizational context and in service of an objective or

goal (after Engeström, 1987). Multitasking inherently reflects the boundaries between ongoing activities; collaboration with particular colleagues often takes place within the context of one or more activities; and the "files" and "piles" used to spatially organize the contents of a workspace are important in classifying information, also closely related to the activities at hand.

Many ongoing research programs seek to understand the role of activity in desktop and ubiquitous computing environments (e.g., Geyer, Vogel, Cheng & Muller, 2003; Dragunov et al., 2005; Tashman, 2006; Bardram, 2007; Kaptelinin & Boardman, 2007; Robertson et al., 2007). Prototype activity-based systems have been shown to match users' real-world work practices more closely than systems based on the traditional application-centric metaphor popular in most mainstream operating systems. It is anticipated that these kinds of systems will provide a variety of benefits to users, including better task awareness, simpler multitasking, more natural organization of information, and improved collaboration.

In this research, I have synthesized empirical data about knowledge work practices (both my own data and those of other researchers), initial findings from the development and evaluation of early activity-based systems (both my own systems and those of other researchers), as well as theoretical understandings of cognition and activity. I have built two early systems to explore aspects of activity-based computing: Kimura, which focused on supporting multitasking and task awareness and exploring interactive, visual representations of activity (MacIntyre et al., 2001), and the sharing palette, which focused primarily on supporting a broad variety of file-sharing practices but also featured an extensible user interface that could easily be adapted to support sharing and collaboration in the context of particular activities (Voida, Edwards, Newman, Grinter & Ducheneaut, 2006). I have also identified a set of key challenges, grounded in theory and practice, for the research and development of activity-based systems: *activities are part of a fluid work practice*, *activities encapsulate evolving work*,

and *activities are collaborative.* In response to these challenges, I designed and implemented a third research prototype, the Giornata system[2]. Through Giornata, I have demonstrated how the traditional desktop metaphor can be re-envisioned to better match knowledge workers' practices by emphasizing activity as a primary organizing principle in GUI-based interaction, information organization, and collaboration. I have deployed a prototype of the system and gathered user feedback on various aspects of the design based on 270 total person-days of real-world Giornata use. I have also evaluated the Giornata system, resulting in an assessment of the technologies that enable and those that pose barriers to the development of activity-based computing systems. Finally, based on my experiences with the Giornata system, I have revisited the challenges in realizing activity-based support for knowledge work.

1.1 Purpose of Research and Thesis Statement

In this dissertation, I address the following questions related to the development of an emerging class of systems aimed at providing activity-based support for knowledge work:

- What are the current mismatches between knowledge work practice and systems support for these practices?
- How do theories of human activity inform the development and anticipate the potential for innovation in activity-based systems for supporting knowledge work?
- In what ways are existing user interface techniques and metaphors sufficient (or insufficient) for representations of and interactions based around activity?
- How do knowledge workers use activity-based systems in the course of day-to-day work?

[2] *Giornata* is Italian for "day's work," and is used to denote both the time during the day that work takes place and, in the context of *buon fresco* (wet plaster) painting, the physical region of a painting that can be completed in a single session.

- What are the technical requirements for these kinds of systems?

By engaging with these questions, I aim to demonstrate the utility of re-examining the desktop user interface through the lens of activity. The following statement summarizes the thesis of this dissertation:

> An activity-centric perspective can drive innovation in desktop computing by guiding the development of new user interface capabilities, metaphors, and techniques. These innovations will be better suited for supporting the multitasking, resource organization, and collaboration practices of knowledge workers than existing computational tools.

1.2 Expected Results and Contributions

Because this research lies at the intersection of several domains of study, including empirical research into knowledge work practices, cognitive psychology and cognitive science, and user interface software and technology, it is my intent to provide a corresponding breadth of contributions to the research community, including:

- a summary of the challenges in supporting activity-based knowledge work, derived from observed knowledge work practices, theories of human activity and cognition, and the affordances of (and obstacles imposed by) underlying technologies and established user interfaces;
- three computational prototypes that address different needs of knowledge workers and that examine novel interface concepts and technology requirements for providing activity-based support for knowledge work; and
- an evaluation of an exemplar activity-based system in supporting the multitasking, resource organization, and collaboration practices of knowledge workers.

1.3 Organization of the Dissertation Document

This dissertation consists of seven main chapters (numbered 2–8), representing three major phases of research:

- assessing the research community's current understanding of knowledge work practice, the role of activity in cognition, and the current state of the art in computing tools for knowledge work,
- enumerating specific challenges for designing activity-based computing tools, and
- a case study of the design and implementation of one such system.

Chapters 2 and 3 include the related work for the dissertation as well as several early empirical and technological explorations into the domain of activity-based computing. Chapter 2 seeks to answer the question "what is an activity?" In this chapter, I provide a survey of previous empirical studies of knowledge work and theories of activity from the fields of cognitive science and cognitive psychology. I also present the results of an empirical study that I conducted, which explored the relationship between knowledge workers' mental models of activity and their window management behaviors on the desktop computer. Chapter 3 presents a complementary survey of previous technological explorations of activity-based computing, including two systems that I built to explore various specific aspects of activity-based computing in depth: Kimura and the sharing palette.

Chapter 4 serves to reflect back over all of the prior research identified in chapters 2 and 3, culminating in the synthesis of three fundamental challenges for the development activity-based computing systems.

Chapters 5 and 6 present a detailed account of how I parlayed these challenges into the design, implementation, deployment, and evaluation of an activity-based computing system grounded in empirical data, theory, and the technical affordances of contemporary desktop computers. Chapter 5 describes the design and implementation of this system, relating various aspects of the design back to the specific challenges that they seek to address. Chapter 6 presents a detailed analysis of the findings from a longitudinal, real-world deployment of Giornata.

Chapters 7 and 8 conclude the dissertation by reflecting back on different aspects of the research. Chapter 7 provides a high-level analysis of the technologies that enabled the realization of the three activity-based computing systems described in this dissertation and suggests areas in which future technology development could help to further enable prototyping and deployment of this class of systems. In Chapter 8, I reflect back on the challenges enumerated in chapter 4 in light of my experiences working with Giornata, re-examining the thesis statement and research questions posed as part of the larger research agenda and looking ahead to future research opportunities in the activity-based computing domain.

Although this structure places a significant emphasis on the third phase of the research agenda, it is not intended to minimize the contributions embodied in the realization of the earlier systems or the identification of the challenges; rather, it is intended to convey the iterative research process and the evolution of my thinking about activity-based systems over the course of the last several years.

CHAPTER 2

WHAT IS AN ACTIVITY? EMPIRICAL AND THEORETICAL GROUNDING FOR ACTIVTY-BASED COMPUTING

Knowledge workers, as a population, have been the frequent subjects of observation and study in the HCI literature. Over the last twenty years, numerous studies have been published detailing typical knowledge work practices from a variety of perspectives. These studies reveal the significance of activities in day-to-day work and help explain how technology has helped, hindered, and been re-appropriated to better support activity in knowledge work. These studies also begin to suggest requirements for future activity-based computing systems in support of knowledge work.

However, studies of existing practice can only reveal the current state of the world and do little to suggest what might happen when activity-based computing tools are actually deployed and adopted. Although iterative design and deployment could be used to adjust the capabilities of the computational tools as the technology and its users co-evolve, another way to anticipate where future activity-based tools might lend the most support is to look to theoretical understandings of activity from the cognitive science and psychology literature. Taken together, observations of current practice and theories of activity serve to more fully describe the potential design space for activity-based computing systems.

In this chapter, I begin with a brief survey of previous empirical studies of knowledge workers as they relate to activity-based computing (adapted from Voida, Mynatt & MacIntyre, 2007). I have clustered these findings into several high-level themes for the purposes of drawing comparisons across similar studies, sometimes carried out years apart and with different user populations, and in the interest of translating the observed practices into somewhat generalizable design implications for

activity-based computing systems. In the second section, I present the findings from an empirical user study utilizing a small activity prediction and experience sampling application to probe the relationship between users' mental models of switching among activities and their window management behaviors on the desktop (adapted from Nair, Voida & Mynatt, 2005). In section 2.3, I transition into a survey of some of the significant theoretical explorations of activity, providing an overview of some of the cognitive and psychological models that have been proposed to explain the ways that people organize their work (adapted from Voida et al., 2007). Finally, I close with a discussion of how these empirical observations and theoretical models might be used as a starting point in the development of activity-based computing systems.

2.1 Empirical Studies of Knowledge Work Practices

Drucker (1974) introduced the term "knowledge worker" to describe an emerging (and quickly growing) role in business organizations:

> The manual worker is yesterday…. The basic capital resource, the fundamental investment, but also the cost centre for a developed economy is the knowledge worker who puts to work what he has learned in systematic education, that is, concepts, ideas and theories, rather than the man who puts to work manual skill or muscle.

Finding this definition somewhat too vague to use as the basis for developing computer-based support for such workers, Kidd (1994) further refined Drucker's definition in a number of ways. She provided more concrete examples of job fields that constitute knowledge work, "design, advertising, marketing, management consultancy, broadcasting, law, finances and research." She suggested four defining characteristics of knowledge work:

- "that they themselves are changed by the information they process;"
- "each knowledge worker produces a different output and it is this variation which is their key benefit to the company;"
- "[a] low dependence on filed information;" and

- "[the] importance of spatial layout and materials" to knowledge workers (Kidd, 1994).

Finally, Kidd constrained the definition of knowledge work by describing what it isn't. She specifically excludes communication workers and clerical workers from her definition, since communication workers primarily serve as "tuner amplifiers for information which they collect from other sources (e.g., knowledge workers, magazines or conferences) and pass on…in order to bring about changes in other people's understanding, beliefs, [and] behaviour rather than in their own," and clerical workers "apply information which is extrinsic to them and which does not change (ie. inform) them, e.g., company policies" (Kidd, 1994).

While Drucker and Kidd provide high-level definitions for knowledge work and knowledge workers, many other studies have explored detailed aspects of knowledge work practices. These studies fall into three general categories: studies of multitasking and interruptions, studies of information organization, and studies of collaboration.

2.1.1 Multitasking and Interruptions

Knowledge work is often associated with the notion of multitasking. At any point in time, knowledge workers (and managers, in particular) are involved in multiple, interwoven activities (Bannon, Cypher, Greenspan & Monty, 1983; Sproull, 1984). These activities tend to be nested within one another, that is, "users rarely complete any time-consuming activity before beginning another task" (Bannon et al., 1983). Furthermore, knowledge workers are generally "proud of their ability to multitask, and they reported feeling that multitasking brought fun and variety to their work" (Czerwinski, Horvitz & Wilhite, 2004).

Given the importance of multitasking to knowledge work, researchers have devoted much time and attention to understanding the constituent activities involved in multitasking behavior. Based on hours of shadow observation and interview-based

fieldwork, González and Mark (2004) provide the following definition for an ongoing activity, a construct they term a *working sphere*:

> We define a working sphere as a set of interrelated events, which share a common motive (or goal), involves the communication or interaction with a particular constellation of people, uses unique resources and has its own individual time framework. With respect to tools, each working sphere might use different documents, reference materials, software, or hardware. It is the whole web of motives, people, resources, and tools that distinguishes it from other working spheres.

This definition succinctly captures various working definitions of *task* and *activity* found elsewhere in the literature and used as a basis for activity-based computing systems.

Interruptions have also been an area of substantial research, which is perhaps unsurprising since interruptions have been cited as the cause for activity switches around 50% of the time (Sproull, 1984; González & Mark, 2004); the statistic has been found to increase to an average of 59% for managers (Mark, González & Harris, 2005).

2.1.2 Information Organization

In her focused description of knowledge workers, Kidd notes several interesting behaviors related to filing and the spatial organization of knowledge workers' information:

- Knowledge workers, in general, have a low dependence on filed information. They tend to take prolific notes, which are important to them in the process of meaning-making but are not typically used as a reference on their own.
- The spatial layout of a knowledge worker's materials is important as a "holding pattern" for short-term organizational purposes, before the materials have been classified and can be filed.
- The spatial layout of a knowledge worker's materials is important as a primitive language, since the physical (and, presumably, digital) artifacts stand in as a model of real-world phenomena.

- The spatial layout of a knowledge worker's materials is important as a contextual cue for resuming a suspended activity; the location of artifacts helps to answer the question, "where was I?"
- The spatial layout of a knowledge worker's materials is important as demonstrable output, since piles in some ways quantify the work that has been accomplished (Kidd, 1994).

Many of these behaviors echo previous findings from Malone's study of knowledge worker's desk organization practices (Malone, 1983). Just as Kidd describes the importance of spatial organization and "holding patterns" for not-yet-classified information, Malone notes the distinction between *files* and *piles* in the office environment:

> Two of the most important units of desk organization are *files* and *piles*. Both files and piles are ways of collecting groups of elements into larger units…. [*F*]*iles* are units where the elements (e.g., individual folders) are explicitly titled and arranged in some systematic order (e.g., alphabetical or chronological)…. In *piles*, on the other hand, the individual elements (papers, folders, etc.) are not necessarily titled, and they are not, in general, arranged in any particular order (Malone, 1983).

His study also serves to support Kidd's claim that information organization is an important part of the meaning-making process: "The difficulty of deciding how to classify something can be an important barrier to filing the information" (Malone, 1983). Lansdale (1998) clarified this point in the context of electronic information systems, noting that "the problem…lies in deciding what categorizations to use, and in remembering later exactly what label was assigned to a categorization."

Barreau and Nardi (1995) encountered many of these findings again in the workplace over a decade later, in an era when electronic storage of knowledge work-related artifacts was much more commonplace. They found, among their study population of desktop computer users:

- a preference for location-based search for finding files (in contrast to logical, text-based search);
- the use of file placement as a critical *reminding function*; and
- the "lack of importance" of archiving files (Barreau & Nardi, 1995, paraphrased by Freeman & Gelernter, 2007).

This process of organizing information to make meaning within activities spans many different kinds of resources, both physical and digital. According to a recent study by Bergman, Beyth-Marom and Nachmais (2006), users tend to think about and classify their personal information in terms of activities more than they do in terms of information type.

Not only do knowledge workers employ spatial layout as a tool for making sense of information resources individually, empirical evidence also suggests the importance of doing so in the context of face-to-face collaboration (Scott, Carpendale & Inkpen, 2004). When working with information objects projected on tabletop displays with collocated colleagues, the spatial positioning of those objects played a significant role in how those present made sense of the information and communicated collaborative intent with one another.

2.1.3 Collaboration

Collaboration is an important component of knowledge work. While it was been reported that for knowledge workers, "deskwork…clearly consumes a main portion of the day [36%]…unscheduled meetings…constitutes the second largest category [18.9%]" (González & Mark, 2004). Additionally, of all the tasks knowledge workers reported being involved in during the course of a diary study, 23% of the tasks reported "could best be described as 'email'" (Czerwinski et al., 2004).

E-mail and collaboration are not just additional tasks that are undertaken during the course of knowledge work; they also serve important roles in helping to manage and

organize activity. Whittaker and Sidner (1996) studied the e-mail practices of knowledge workers and observed that, for their participants, task management was largely a matter of "[ensuring] that information relating to current tasks is readily available." Maintaining a well-organized e-mail inbox was both central to task management and a means for knowledge workers to keep themselves organized.

The importance of collaboration increases when the knowledge worker also serves in a managerial role. Panko (1992) summarized several observations about managers' collaboration practices:

- managers spend 25-60% of their time engaged in communication;
- the majority of a manager's meetings are dyadic face-to-face encounters; and
- the higher a manager in the corporate hierarchy, the more time he or she spends in meetings of one form or another.

Haythornthwaite, Wellman & Mantei (1995) also studied the diversity of communications channels that are utilized during the course of a typical work day and found that those individuals with a management-level role in an organization (those engaged in the dimension of "giving work") relied extensively on both unscheduled meetings and e-mail to accomplish their goals.

If collaboration is an important part of knowledge work, then contact management must necessarily be an important component of knowledge work as well. Whittaker, Jones and Terveen (2002) performed an analysis of knowledge workers' contact information stores and communication histories and found that the average number of contacts represented in these stores and histories was over 600. However, they also determined that their study participants only considered an average of 119 of these contacts as "important."

Finally, information exchange is reported as an important aspect of collaboration for knowledge workers. Of Ducheneaut and Bellotti's more than 60 respondents, all but one reported regularly using e-mail to exchange files (Ducheneaut & Bellotti, 2001).

2.2 An Empirical Study Exploring Activity Switches and Window Management Behavior

There is a significant body of research that shows that users are very particular about the layout of windows on their desktop. As part of their work on the Rooms virtual workspace, Henderson and Card (1986) suggested that users have a "working set" of windows active at any given time. By moving between the windows of the tools (applications), users are implicitly defining this working set of windows. Studies of the window management behavior of computer users found distinct differences in visible windows when users switched between activities when compared to window switching within a single activity (Hutchings & Stasko, 2004; Hutchings et al., 2005). Users were as interested in hiding windows irrelevant to their current activity as they were in displaying (and working with) relevant ones. Within-activity switching was also found to involve fewer window manipulations than a switch between major activities.

When a user switches between activities, she opens the necessary windows to her favorite positions—this is not restricted to opening new windows and can occur even if she is merely rearranging pre-existing windows into a new configuration. The hypothesis explored in this study is that the action of switching between activities produces window operations at a different frequency than does normal work. The study tests this hypothesis using software to combine low-level observations of window manipulations and their frequency distribution to determine when an activity switch might be taking place, and comparing these inferred activity switch instances to users' self-reported activity switches.

2.2.1 System Design

The application used in the study runs in the background and uses window manipulation frequencies to determine when an activity switch has occurred. The program tracks window events such as *create, activate, maximize, minimize, hide, show*

and *destroy* by programmatically hooking into window system events that are publicly available on the Microsoft Windows operating system. This event information is time-stamped and augmented with details of the id, name, caption and class of the generating window before being sent to the detection algorithm and added to a log file.

2.2.1.1 Detection Algorithm

The activity switch detection algorithm begins by filtering the window events so that only the events associated with top-level windows are used. The timestamps are then used to calculate the average time between window events for a given activity. The system also calculates the simple moving average of the time between the last k interactions. When the ratio between the overall activity average and the moving average exceeds a certain threshold, the system generates a new activity event and resets the averages.

Example: Consider a situation in which the algorithm is evaluating the $(n+1)^{th}$ window event. t_i is the time between the i^{th} and $(i-1)^{st}$ event:

$$\text{Activity average, } v_{AA} = \frac{1}{n}\sum_{i=1}^{n} t_i \qquad (2.1)$$

$$\text{Moving average, } v_{MA} = \frac{1}{k}\sum_{i=n-k}^{n} t_i \qquad (2.2)$$

$$\text{Ratio, } r = \frac{v_{AA}}{v_{TA}} \qquad (2.3)$$

Pilot testing was used to determine that a moving average of the last 7 windows ($k = 7$), coupled with threshold values of 0.67 (lower limit) and 1.5 (upper limit), provides a good compromise between accurate activity switch detection and the avoidance of false positives. Further improvements may be possible by using an exponential weighted average or by dynamically altering the threshold and k values based on past performance.

2.2.1.2 User Interface

Once the system detects an activity switch, it displays a small pop-up window in the bottom-right corner of the screen (Figure 2.1). This pop-up asks the user if he has changed activities and, in the case of a switch, what that new activity is. If the user confirms the switch, the system registers it as a successful detection and resets its averages to detect the next activity. If the user cancels or ignores the pop-up, the system registers it as a false positive. If there is no interaction from the user, the pop-up automatically disappears after 20 seconds and the program tries to detect the next activity switch without resetting its averages. In order to minimize the disturbance to the user, the system was designed to display no more than one pop-up window every 5 minutes. The choice of the 5-minute interval was based on the belief that most users' activities would last for longer than 5 minutes (after González & Mark, 2004), with the hope that this limitation would allow the system to detect most of the activity switches while simultaneously ameliorating user irritation due to frequent interruptions by the software.

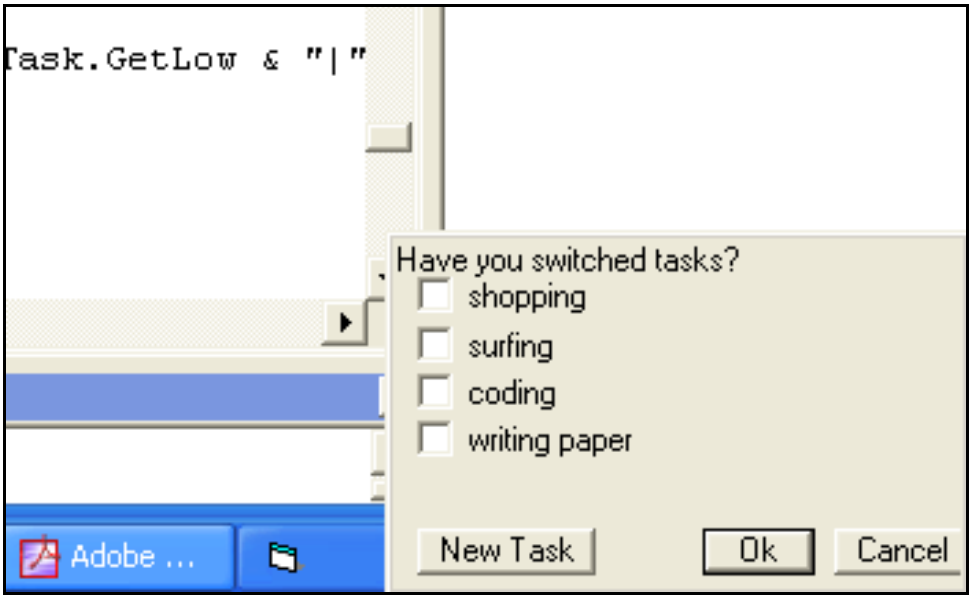


Figure 2.1 A screenshot of the application's activity switch confirmation pop-up window.

2.2.1.3 Reinforcement Learning

Since users reported what activity they were performing, the system could utilize reinforcement learning to tune the threshold limits of the detection algorithm for a given activity. If the system correctly identified an activity switch, a positive reinforcement was applied, reducing the threshold values for detection; for false positives, the threshold value was increased. The amount of feedback applied was inversely proportional to the number of past reinforcements the system received; this created damped oscillations in the threshold values before settling around the optimal values.

2.2.2 Study Design

The software was evaluated by deploying it to six participants for a period of two weeks. The participants consisted of two professors (P1 and P2), three graduate students (G1, G2, and G3) and one IT professional (IP). All the participants were experienced computer users and used Microsoft Windows as their primary operating system. The software was installed on their primary machines for a period of two weeks, during which time they were asked to enter the name of their current activity whenever the software correctly detected an activity switch. The participants were told about the reinforcement learning and were encouraged to train the system by giving it feedback. Once the study was complete, the participants were interviewed about their activity management styles and the software's effectiveness in picking out activity switches.

2.2.3 Findings

Of the six participants, all participated for a full two weeks (in some cases voluntarily extending participation to over 3 weeks). By the end of the study, 1063 activity switches had been detected across all six users, with 422 (39.70%) being confirmed as accurate (Table 2.1). However, since the duration of participation varied

across users, a more representative measure of the system's accuracy would be the average accuracy, computed across all users (49.70%).

Table 2.1 Summary of activity switch detection accuracy.

Participant	Activity Switches Detected	Activity Switches Confirmed	Accuracy (%)
Professor 1 (P1)	57	54	94.74
Professor 2 (P2)	76	43	56.58
Graduate Student 1 (G1)	367	123	33.51
Graduate Student 2 (G2)	217	117	53.92
Graduate Student 3 (G3)	91	37	40.66
IT Professional (IP)	255	48	21.33
Total	**1063**	**422**	**39.70**

Participants P1 and P2, who perform similar duties in their work, had activity detection accuracies of 95% and 57%, respectively. P1 was very appreciative of the accuracy and said that the software was able to find almost all of her activity switches. P2, on the other hand, was concerned that the software was unable to detect activity switches if he switched between activities very quickly. He also noted that there were occasions in which the software erroneously detected activity switches when he switched between windows common to a larger, single activity. The lower accuracy for P2 was likely due to a combination of his fast activity switching and the 5-minute time-out programmed into the software; since P2 would often switch activities within 5 minutes, the pop-up would appear only after the 5-minute timeout had expired.

The accuracy of detection for G1, G2 and G3 was 34%, 54% and 41%, respectively. The relatively low accuracy for these participants can be partially explained by their use of instant messaging (IM) software. Unlike P1 and P2, these participants were regular IM users and would often chat on IM while concurrently working on their primary activity. This complex multi-tasking behavior, along with the bursty nature of IM

conversations, led to several activity-switch notifications being generated due to IM windows. In most of these cases, the participants would often either cancel or ignore the pop-up because they did not perceive IM use to be a separate activity.

Participant IP was an entry-level IT professional who primarily wrote and documented software code. While the window tracking system gave IP the lowest accuracy (21%), he was also its most enthusiastic supporter. In fact, IP was so impressed with the accuracy of the switch detection that he went on to use the log files of the detected activity switches to fill out his weekly project time sheets. When questioned about the system, IP said that while it did not miss any activity switches, it would sometimes be triggered while switching between windows of sub-activities.

2.2.4 Discussion

Despite the wide variation in accuracy, this study demonstrated the feasibility of using a window manipulation frequency-based approach to detecting activity switches. One of the most important things that learned as a result of this study is that the system must adapt to users' work habits: the observed variation in detection accuracy can be explained, to a large extent, by differences among the study participants' work patterns. For example the 5-minute time-out adversely affected only P2 since he switched between activities much more quickly than the other users. Participants G1, G2 and G3 were primarily affected by IM windows' propensity to appear at unusual times and co-exist alongside other windows more central to the primary activity. These participants asked to have IM excluded from the window tracking. When questioned about their IM use, these participants felt that they were not unduly hampered by the notifications; this is contrary to the finding reported by Czerwinski, Cutrell & Horvitz (2000), who showed that IM message notifications have a reliably harmful effect on the primary activity.

Currently, the system cannot distinguish if a user moves to a different activity within the same window. The most common example of this problem involves the use of

web browsers, where users may use the same window for anything from online banking to checking e-mail. One possible extension of this work would be to incorporate URL information in future algorithms to detect activity switches based additionally on web browser usage. Another area of future work also includes tailoring the window tracking to suit individual users and their work habits; possibilities include allowing the system to filter out specific windows (such as IM) and dynamically altering the size of the moving average window and the pop-up window's time-out based on user feedback. Dynamic adaptability may also allow this technique to track when users change between sub-activities (Iqbal, Adamczyk, Zheng & Bailey, 2005) instead of only considering independent, top-level activities.

2.3 Theoretical Explorations of Activity

In the process of trying to understand mental processes and cognition in general, researchers in the fields of cognitive science and cognitive and social psychology have advanced various theories of human activity. These theories build upon lower-level theories of phenomena such as memory, attention, and motivation, and represent a variety of points of view on the correct unit of analysis for a concept that is cognitively quite complex and involves processes both internal and external to a single human actor.

These theories have been utilized at different times by HCI researchers seeking to understand why particular systems succeed or fail and to construct systems that match (as closely as they can) the mental processes of the user. Activity theory is one of the more widely used examples of these theories, since it explicitly deals with modeling and representing activity and captures notions of the social context in which work takes place (see Nardi, 1996). Much of my work has been focused at the intersection of activity theory and reported or observed knowledge work practice, although there are points at which other theories can also inform the design of activity-based systems.

2.3.1 Activity Theory

The origins of activity theory can be traced back to the former Soviet Union as part of the cultural-historical school of psychology founded by Vygotsky, Leont'ev, and Luria. Rather than focusing on *action* as a unit of analysis, activity theory focuses at the broader level of an *activity* and incorporates the social and cultural context of cognition (Halverson, 2001; Leont'ev, 1978; Vygotsky & Cole, 1978).

Engeström (1987) provides a classic visualization summarizing the structure of an activity (Figure 2.2). This model is based on three mutual relationships: that between the actor (*subject*) and the community (other *actors involved*), that between subject and the object (in the sense of *objective*) of the activity, and that between the object and the community:

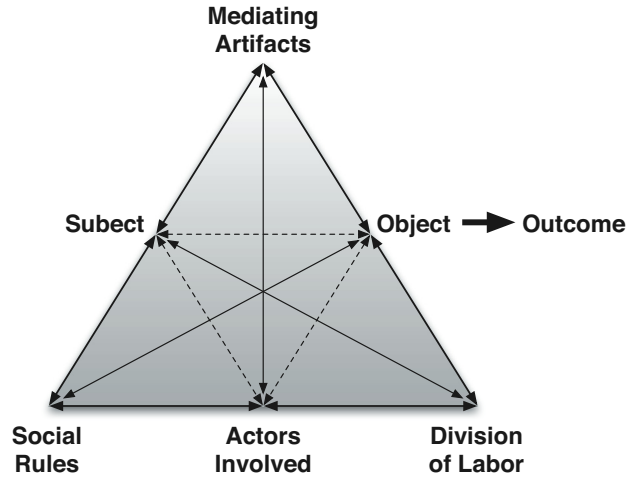


Figure 2.2 An adaptation of Engeström's analysis of activity and mediating relationships.

These mutual relationships are *mediated* by the other components of activity. For example, the relationship between subject and object is mediated by tools (*mediating artifacts*); because of this, the subject's experience of the object is constrained by the tools used, and the tools that are created as a by-product of the activity are directly

shaped by the subject and the object. The tools also embed the culture and history of the other components of the activity, such as the social rules governing the community, the community itself, and the organization of that community (e.g., the roles of its members), sometimes referred to as the *division of labor*.

However, Gay and Hembrooke (2004) point out a weakness in Engeström's formulation of activity theory: "The model of activity theory…has traditionally been understood as a synchronic, point-in-time depiction of an activity. It does not depict the transformational and developmental processes that provide the focus of much recent activity theory research."

Boer, van Baalen, and Kumar (2002) provide a proposal for how the scope of activity theory can be expanded across time and the levels of an organization to explain connections between different activities as well as the influence that an activity may exert upon itself:

> Besides the fact that an activity is situated in a network of influencing activity systems, it is also situated in time…. In order to understand the activity system under investigation, one therefore has to reveal its *temporal interconnectedness*…. Rather than analyzing an activity system as a static picture of reality, the developments and tensions within the activity system need to be described and analyzed…. When analyzing an activity system at a particular contextual level, one should also take into account its relations with activity systems at other contextual levels (e.g., economic system, industry, supply chain, organization, department or production process)…. The activity system under investigation is not only affected by activity systems at other contextual levels, it also exerts influence on them itself…. This is in line with Giddens' theory of structuration which assumes that on the one hand human action is restricted by institutional properties of social systems, while on the other hand these institutional properties are the product of human action (Boer, van Baalen & Kumar, 2002, authors' emphasis).

Boer et al. also consider the role that an activity may play in other activities at different levels of analysis. They suggest that the components of one activity system may play different roles in more broadly- or narrowly-scoped activities that exist in different cultural contexts, for example, on a project team, in a department, or in an entire

corporation (see Figure 2.3). This approach provides activity theory with a similar degree of agility in representing complex, cooperative activity as alternative theoretical approaches, such as distributed cognition (Hutchins, 1995; Kirsh, 2001).

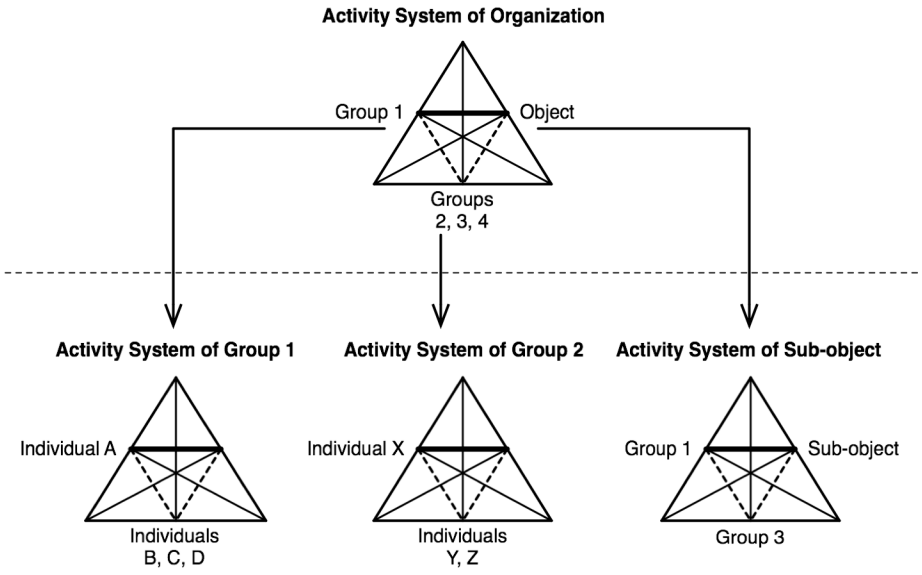


Figure 2.3 Relationships among different levels of analysis (after Boer, van Baalen & Kumar, 2002).

Kaptelinin and Nardi suggest an additional enhancement to the traditional activity theory model to fill an apparent gap between very high-level activities and low-level actions, a preliminary construct they term *engagements* (2006). Engagements represent short-term, focused blocks of "actual work" that allow the actor to ignore some of the long-range objectives that might otherwise distract them (or, in the worst case, paralyze them) from accomplishing concrete and immediate tasks. This construct is intended to better map to the granularity of "working contexts" (MacIntyre et al., 2001) that activity-based computing systems are commonly designed to support.

2.3.2 Situated Action

Suchman (1987) provides a complementary model for human activity in situated action. In contrast to the somewhat rigid, structured models of activity suggested by theorists and instantiated in the workflow approach to collaboration support popular at the time, she emphasized the importance of differentiating between work and representations of work-like plans and process models.

The theory of situated action views of human activity as a series of situated and ad hoc improvisations, which makes plans rational anticipations before the activity is undertaken and post hoc reconstructions afterward. Bardram characterizes this view as what he calls a "planning paradox":

> On the one hand, due to the contingencies of the concrete work situation work has an ad hoc nature. Plans are not the generative mechanisms of work, but are 'merely' used to reflect on work, before or after. On the other hand, we find that plans, as more or less formal representations, play a fundamental role in almost any organisation by giving order to work and thereby they effectively help getting the work done (Bardram, 1997).

This view of the planning paradox suggests that representations of activity serve different purposes at different times—that flexibility at the moment of action is essential, but that structured records of activity can serve as important organizational, communicative, and collaborative artifacts as well. Bardram uses this argument as one of the foundations for his take on activity-based computing. He combines aspects of activity theory and situated action by treating plans—that is, models of activity—as mediating tools produced by and used within activities:

> Plans as artifacts are used to mediate activity regardless of whether they exist on e.g. paper or are memorised. Human work is characterised by the collaborative production of artifacts; each made with the purpose of mediating a certain activity. The mediating characteristics of an activity is therefore *crystallised* (or objectified)…into these artifacts, and through use, the artifacts are continuously modified and shaped to meet the evolving human needs…. Cognitive plans and their material counterpart are mere reflections of each other because they are both resources for, and products of, human activity (Bardram, 1997).

2.3.3 Psychological Theories of Activity

A third theoretical approach to understanding human activity comes from the field of cognitive psychology. Miyata and Norman (1986) suggest that a "psychology of multiple activities" can be derived from empirical studies of memory (short-term memory and working memory), studies of attention, and studies of action. While their approach differs from the others presented in this section, many of their conclusions are similar. They categorize activities as being one of three types: foregrounded (under conscious control), backgrounded (either delegated to an external entity or ongoing under automatic or unconscious control), and suspended.

Echoing the findings from Malone's study of knowledge workers' desk organization practices (Malone, 1983), Miyata and Norman (1986) claim that spatial location is a kind of memory aid that minimizes the need for cognitive processing of icons or text.

CHAPTER 3

TECHNOLOGICAL EXPLORATIONS OF ACTIVITY-BASED COMPUTING

When considered together, empirical studies of knowledge workers and theories of human cognition can be used to characterize the role of activity in knowledge work, which can then be used as a starting point for designing computational tools in this domain. Another approach that can also be instructive is to consider some of the systems that have already been designed and implemented—and, in many fewer cases, deployed—for supporting some of the same facets of knowledge work, including multitasking, task awareness, information organization, communication, and collaboration. This kind of complementary analysis of prior technological explorations provides a pragmatic lens for activity-based systems research, revealing not only the breadth of design decisions that have been exercised in the past, but also the assumptions about activity and knowledge work embodied in these designs, the technological hurdles that have influenced the implementation of these past systems, and the challenges that the realization of these computational artifacts have uncovered.

In this chapter, I present a survey of previous technical explorations in activity-based computing, including both those systems developed by others in the human-computer interaction research community and two that I designed and built in earlier phases of this research: Kimura and the sharing palette user interface. Kimura is designed to explore computational support for multitasking and task awareness using a novel combination of primary and peripheral projected displays. The sharing palette is a lightweight user interface for sharing files, designed in response to real-world file sharing practices and breakdowns. These systems are each intended to explore, in depth, different aspects of knowledge work; both provide additional insight into providing tool-level

support for this domain as well as additional data about knowledge work practices, themselves.

3.1 Existing Research in Activity-Based Computing

Table 3.1 summarizes prior research systems within the domain of activity-based computing, including the goals motivating the development of each and some of the assumptions embodied in each system regarding activity and activity representations.

Table 3.1 A summary of other researchers' previously-reported technical explorations into the design of activity-based computing systems and the contributions and assumptions of those systems.

Activity-Based Computing System	Key Research Contributions and Design Assumptions
Rooms (Henderson & Card, 1986)	Established the paradigm of virtual desktops. System design was based on the "window working set" theory of information management described by Card, Pavel and Farrell (1984).
Elastic Windows (Kandogan & Schneiderman, 1997)	Implemented a window manager using a hierarchical, space-filling tiled layout to improve individual window management performance. Also integrated notions of role-based representations at the level of the window management system.
Haystack (Adar, Karger & Stein, 1999; Karger, 2007)	Created an architecture and information management portal based on "semistructured data"—in large part, semantic Web documents and RSS feeds. Some of the key goals of the system included emphasizing use of metadata, encouraging view independence and information aggregation.
Manufaktur (Büscher, Mogensen, Shapiro & Wagner, 1999)	Designed a 3-dimensional workspace manager offering spatial arrangement of documents and references to resources stored elsewhere on a computer network. Users could organize documents into activity-based clusters by taking advantage of spatial organization capabilities.
CPN2000 (Beaudouin-Lafon & Lassen, 2000)	Explored the ability to group application windows ("pages") into user-defined groups ("binders") that behaved as a single window.
Task Gallery (Robertson et al., 2000; Robertson et al., 2007)	Replaced the Windows Explorer interface with a 3-dimensional desktop environment where windows could be grouped and positioned on virtual walls. The interface was designed to evoke spatial memory and cognition in supporting task management.

Table 3.1 (continued)

Activity-Based Computing System	Key Research Contributions and Design Assumptions
TaskView (Gwizdka, 2002)	Re-designed the e-mail client to incorporate spatial, temporal, and task-based visualizations of computer-mediated communication.
Miramar (Light & Miller, 2002)	Iterated on the design of a 3-dimensional file-management environment allowing documents to be clustered together in semantically meaningful, spatial groups as an alternative to the visually homogenous, hierarchy-oriented structures present in traditional file managers.
ContactMap (Nardi, Whittaker, Isaacs et al., 2002; Fisher & Nardi, 2007)	Implemented a computer-mediated communication and personal information management application based on a social network visualization. Activities were effectively organized by the people they involved, building on behaviors already seen in organizing work around/through e-mail.
Aura project (Sousa & Garlan, 2002)	Framework for supporting context awareness and activity-based information organization in a ubiquitous computing environment. A primary focus of the project was in supporting user mobility.
Taskmaster (Bellotti, Ducheneaut, Howard & Smith, 2003)	Fundamentally re-cast e-mail as a task-management activity and provided tools for managing e-mail accordingly. The primary unit of organization in the client was the "thrask," a threaded, task-centric collection of messages, attachments, and links.
UMEA (Kaptelinin, 2003; Kaptelinin & Boardman, 2007)	A system based on the personal information management metaphor for organizing communications, documents, and other resources into project-related pools. The system integrated closely with Microsoft Office applications and provided a suite of typical personal information management tools (e.g., to-do lists, calendars, and notes) for managing ongoing projects.
GroupBar (Smith et al., 2003; Robertson et al., 2007)	Extended the capabilities of the Microsoft Windows TaskBar to semantically group and allow management of multiple windows at the same time.
ActivityExplorer (Geyer, Vogel, Cheng & Muller, 2003; Muller, Geyer, Brownholtz, Wilcox & Millen, 2004)	Client-server application designed to "combine the lightweight and ad hoc characteristics of e-mail and the rich support for sharing and structure in shared workspace systems." Implementation centered on notions of object-centric sharing and awareness, threaded conversational structure, and dynamic group membership.
Sphere Juggler (Morteo, González, Favela & Mark, 2004)	Aggregated and organized resources affiliated with ongoing activities in a hierarchical, list-based user interface for efficient information retrieval and task switching. Categories of stored resources included documents, contacts, e-mails, and "pending issues."

Table 3.1 (continued)

Activity-Based Computing System	Key Research Contributions and Design Assumptions
BumpTop (Agarwala & Balakrishnan, 2006)	Investigated grouping behaviors similar to those provided for windows via virtual desktops, but did so at the level of managing the iconic representations of documents and applications where they are stored.
Scalable Fabric (Robertson et al., 2004; Robertson et al., 2007)	"Focus+context"–based window management software that used the central portion of a computer display for focal interaction with an activity-based group of windows and the periphery to display montage-like representations of background tasks.
Activity-Based Computing (Bardram, 2005; Bardram, 2007)	Fieldwork-inspired system for supporting activity-based collaboration in a hospital environment. The design focused on enabling mobility within a workplace and using multiple devices, sharing objects and activities, providing quick activity suspension and resumption capabilities, and integrating awareness tools with everyday work.
TaskTracer (Dragunov et al., 2005; Stumpf et al., 2005)	Desktop interface for labeling open windows as belonging to an activity and aggregating resources on an activity-by-activity basis for quick retrieval and task resumption. Recent approaches included the use of machine learning techniques to automatically assign user interface actions to activities.
Unified Activity Management (Harrison, Cozzi & Moran, 2005; Moran, Cozzi & Farrell, 2005)	Developed a detailed, flexible model of activity, which was hosted on a central server, including significant emphasis on generalizing activities into templates that could serve as scaffolding for future activities. Users interacted with the system using an AJAX/DHTML-based web interface.
TaskZones (Hutchings et al., 2005)	Window management system combining the benefits of a virtual desktop manager with techniques appropriate to window management on a multiple-monitor workstation.
WindowScape (Tashman, 2006)	A window manager that uses a photograph metaphor for lightweight, post-hoc activity management. The interface allows user-specified and system-observed window groupings and allows windows to exist in multiple groups simultaneously.

The goal of supporting multitasking in knowledge work is not new and has received considerable attention in a variety of research communities. Awareness of the need to support multiple simultaneous activities drove the development of the multi-windowed graphical user interface (Card, Pavel and Farrell, 1984; Miyata & Norman,

1986) and the subsequent addition of multiple "virtual desktops" to these interfaces, beginning with Henderson & Card (1986).

Unfortunately, these graphical user interfaces did not provide effective support for managing multiple working contexts. Limited screen real estate made it impossible to maintain an awareness of background activities. Moreover, constraining the interaction to the desktop was a poor match for common human behaviors such as using large amounts of physical space to simultaneously organize, monitor, and manage multiple activities (Mynatt, 1999).

It has long been recognized that a fundamental problem of desktop computer systems is the small amount of display real estate available to users. Starting with Rooms (Henderson & Card, 1986), and continuing though recent 3D systems, such as the Task Gallery (Robertson et al., 2000) and WindowScape (Tashman, 2006), virtually every window-based desktop computer system has had one or more "virtual desktop managers" to help users manage large numbers of application and document windows in the small space of the desktop monitor. The mismatch between the small amount of screen space and the common "messy desk" work practices people engage in when working with paper is argued eloquently by Henderson and Card (1986), and their arguments and observations have formed the basis for most of the subsequent virtual desktop managers.

Rooms was based on the observation that, when working on a specific task, users typically interact with a small "working set" of documents and tools. The difficulties of working on multiple tasks cannot be overcome by simply giving the user more desk (screen) space, since some windows are shared between tasks, making it impossible to arrange the windows so that all windows for all tasks will be near each other. Furthermore, it is difficult to navigate efficiently among window groupings in a large flat virtual space without additional navigation metaphors or constraints.

The "rooms" metaphor allowed users to collect the windows representing these documents and tools into screen-sized rooms, one for each task, and navigate between the

rooms to switch their working set of windows. Rooms, and all subsequent systems, provided a variety of tools for navigating between rooms, obtaining an overview of the space, and sharing windows between one or more rooms (e.g., clocks, system monitors, and control panels). Rooms also allowed a shared window to have a different size and configuration in each room, a feature not found in most subsequent systems.

Over the last several years, a number of research systems have investigated different extensions to and adaptations of the original Rooms virtual desktop metaphor. GroupBar (Smith et al., 2003; Robertson et al., 2007) and Scalable Fabric (Robertson et al., 2004; Robertson et al., 2007) explored the benefits of integrating window grouping and "focus+context" window management into the existing Microsoft Windows desktop interface. GroupBar's TaskBar-based window grouping functionality allowed users to create semantically-meaningful clusters of windows and manipulate them as a single unit, similar to the idea of a Rooms workspace. However, because the system also drew heavily from the established Windows interaction paradigm, it did not require adoption of a completely new window manager. Scalable Fabric divided the screen into a focal region (in the center) and a peripheral region (around the edges), allowing users to drag windows into the periphery to create Kimura-like clusters of windows representing background tasks. WindowScape (Tashman, 2006) provided virtual window management tools that were driven both implicitly, though automated observations of window focus changes over time, and explicitly, based on users' commands to store a particular window configuration for quick access in the future. TaskZones (Hutchings et al., 2005) explored interfaces for managing virtual desktops on systems that used multiple monitors.

Other activity-oriented window management approaches took a dramatically different perspective from the "virtual desktop" metaphor pioneered by Rooms. Instead of segmenting overlapping windows into semantically related groups, Elastic Windows (Kandogan & Schneiderman, 1997) adopted a space-filling strategy for displaying windows and managing activities. The system's "treemap"-style visualization provided a

hierarchical structure for the display of windows: the containers at the highest level corresponded to the user's different "roles" (e.g., "university research and teaching, industrial, and personal"), the windows at the next level represented the activities carried out within the context of each role, and the actual artifacts and document windows associated with each activity were nested at the next level. In order to switch from activity to activity using Elastic Windows, the user would simply maximize the appropriate sub-tree in the window hierarchy and the desired activity would stretch to fill all available space on the screen.

There have also been attempts to leverage our 3D abilities within a standard display by replacing the 2D desktop with a 3D world containing 2D documents (e.g., Büscher, Mogensen, Shapiro & Wagner, 1999; Robertson et al., 2000, Light & Miller, 2002). Of these, Task Gallery (Robertson et al., 2000; Robertson et al., 2007) has gone the furthest in bringing live 2D applications into a true 3D world. This system took advantage of the input and rendering redirection features of a custom version of Windows 2000 to present existing 2D applications in a 3D space. The Task Gallery was effectively a 3D version of Rooms, where the rooms were laid out along a 3D hallway, with the current room on the wall at the end of the hall. While proposing a number of interaction metaphors to support interacting with 2D documents in a 3D environment on a 2D display, the Task Gallery still suffered from many of the same limitations of Rooms, stemming from the lack of screen real estate.

Manufaktur was a 3D collaborative workspace supplement to the traditional 2D desktop that used OLE/ActiveX containers to capture images of live applications on the Windows desktop (Büscher, Mogensen, Shapiro & Wagner, 1999). It focused on supporting the organization of documents in a shared 3D world, analogous to how designers and architects organize physical artifacts in the real world. Using the system, users could select documents and models for inclusion in the 3D workspace, arrange them according to their working contexts, and reopen them at a later time. However,

Manufaktur was not specifically designed to support multitasking activities, being analogous more to a file manager than a task manager. The Miramar system (Light & Miller, 2002) shared many of the same goals—and limitations—as Manufaktur.

Contemporary systems have augmented the virtual desktop metaphor with much more explicit representations of activity. TaskTracer (Dragunov et al., 2005; Stumpf et al., 2005) is one such system, and was one of the earliest examples of an activity-based computing system designed to manage both windows and digital artifacts associated with ongoing activities. TaskTracer augmented window title bars with a pull-down menu, which could be used to indicate the activity with which each window should be associated. The system also featured deep integration with various applications (e.g., Internet Explorer, Microsoft Office) to track a variety of user interface events (e.g., open, close, save, cut/copy/paste, and sent e-mail). One of the long-term goals for the TaskTracer project was developing a rich user observation platform to inform the development of machine learning algorithms for automatically classifying activities.

Although much of the early work in activity-based systems sought to address the window management issues that emerged due to the widespread adoption of graphical user interfaces, another computing domain where activity has featured prominently is in the design of novel personal information management tools. Generally speaking, these systems have utilized the concept of activity for the purpose of helping the user to organize their disparate files, contacts, and electronic communications. Haystack (Adar, Karger & Stein, 1999; Karger, 2007) was one of the first such systems to radically re-examine the metaphors used to manage personal information, prioritizing temporality as its primary organizational lens. ContactMap (Nardi, Whittaker, Isaacs et al., 2002; Fisher & Nardi, 2007), grounded in studies of e-mail and instant messenger use, based its information representations around the user's social networks; activities in this system could be inferred by the group of colleagues with whom information had been shared over time. Taskmaster (Bellotti, Ducheneaut, Howard & Smith, 2003) proposed a model

for visualizing and managing a user's e-mail inbox based on the concepts of conversational threading and tasks.

Several of the more sophisticated systems, including UMEA (Kaptelinin, 2003; Kaptelinin & Boardman, 2007), ActivityExplorer (Geyer, Vogel, Cheng & Muller, 2003; Muller, Geyer, Brownholtz, Wilcox & Millen, 2004), and Sphere Juggler (Morteo, González, Favela & Mark, 2004), aimed to create entirely new personal information management portals that could serve as activity hubs. Although these systems were not integrated directly into the desktop, they were designed to serve as a central point of interaction for managing projects, organizing and filing information artifacts, receiving and replying to electronic communications, and launching secondary applications when needed. Unified Activity Management (Harrison, Cozzi & Moran, 2005; Moran, Cozzi & Farrell, 2005) represents perhaps the most evolved example of this class of systems, featuring an extremely detailed and extensible model of activity, powerful online collaboration tools, and the ability to create activity templates based on abstractions of ongoing activities. This last capability is particularly significant, because these activity templates are intended to encourage colleagues to share their organizational and procedural knowledge with one another using activity-based representations that can be quickly instantiated and appropriated in a variety of different scenarios.

A third class of activity-based systems has focused on extending awareness of activities off of the desktop and into the "real world." Activity-Based Computing (Bardram, 2005; Bardram, 2007) was a framework and user interface grounded in observations of work in a hospital setting. This system focused on supporting mobile work across multiple devices and fostering collaboration in highly dynamic work environments. The Aura project (Sousa & Garlan, 2002) incorporated representations of activity as the core of a system intended to serve as a proxy for a busy, mobile user working in an environment filled with ubiquitous computing technology. Both of these

research efforts resulted in the creation of entirely new application development frameworks for writing activity-based computing applications.

3.2 The Kimura System

The Kimura system explores how activity models, peripheral displays, and context-awareness could be used to support task awareness and multitasking in knowledge work (MacIntyre et al., 2001; Voida, Mynatt, MacIntyre & Corso, 2002). This research takes as a starting point the use of interactive, projected displays in individual offices. Often discussed in the context of ubiquitous computing and augmented environments, these displays are envisioned as a natural extension to traditional computing in a work setting. In particular, Kimura is designed to leverage projected displays as peripheral interfaces that complement existing focal work areas and support the natural flow of work across these two settings.

Kimura uses these peripheral displays to assist users in managing multiple "working contexts"—coherent sets of tasks typically involving the use of multiple documents, tools, and communicative artifacts with others. The goal of this research is to leverage large, projected, interactive surfaces to support innate human abilities such as peripheral awareness and human cognitive practices such as multitasking and off-loading information into the physical environment (Hutchins, 1995).

The Kimura system separates the user's "desktop" into two parts, the focal display on the desktop monitor and the peripheral displays projected on the office walls, as shown in Figure 3.1. As the user shifts between working contexts, background activities are illustrated as visual *montages* on the peripheral display.

Figure 3.1 The Kimura system, including a desktop component, two interactive peripheral displays with electronic whiteboard capabilities, and a third, non-interactive peripheral display.

From Kimura's point of view, a working context is the cluster of documents related to a general activity, such as managing a project, participating in a conference, or teaching a class, as well as the collection of ongoing interactions with people and objects related to that activity. Any cluster can have numerous documents, including text files, web pages, and other application files, that have been used in the course of the activity, plus indications of ongoing activity such as e-mail messages without replies and outstanding print jobs. Kimura automatically tracks the *contents of a working context*, tagging documents based on their relative importance. As in previous systems, such as Rooms (Henderson & Card, 1986), users demarcate the boundaries of working contexts manually, as this operation is lightweight from the user's perspective and error-prone if left to the system. One contribution of this work is creating and using logs of activity to support both awareness and resumption of background tasks.

Background activities (working contexts) are visualized as a *montage* of images garnered from the activity logs. These montages are analogous to the "room overviews" provided by other multi-context window managers, but where these systems show the

exact layout of the current windows in each room, Kimura's goal is to show visualizations of the past activity in the context. These visualizations help *remind the user of past actions* (see Figure 3.2); the arrangement and transparency of the component images automatically creates an icon for the working context. Another contribution of this work is the design of these visualizations of past activity.

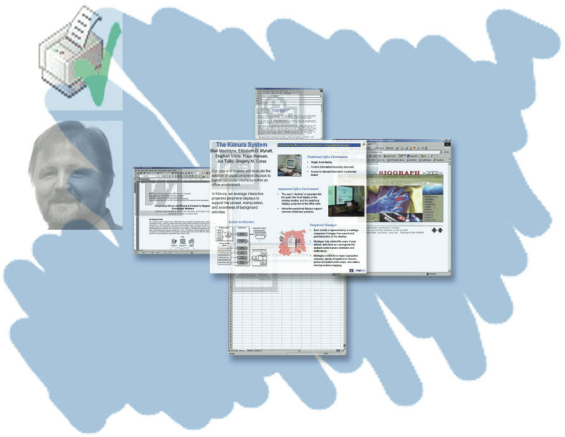


Figure 3.2 A high-resolution rendering of a montage design. Application windows in this working context spiral out from the center based on relative importance. The montage also includes two context notification cues, representing both virtual (completion of a print job) and physical context information (the availability of a colleague).

The montages are displayed on an interactive projected surface and thus help *support common whiteboard practices* (Mynatt, 1999). Users can reposition montages, for example, to indicate the respective priority of background activities, as well as annotate them with informal reminders. Additionally, montages serve as anchors for *background awareness* information that can be gleaned from a context-aware infrastructure. Supporting interaction with the montages and their integration with background contextual cues represents another key contribution of this research.

3.2.1 Scenario

As Charlie walks into his office Monday morning, his whiteboard displays multiple montages consisting of documents and other computer images. Glancing at the board, Charlie decides that working on the new budgets can wait until Wednesday and jots a quick reminder on the montage. Next, he decides to start his day by working on his advisory board briefing for next week. As he selects the montage, his desktop reconfigures to contain the applications he left running when he worked on the briefing last Friday, and the montage appears on the wall near his monitors. The Netscape browser still contains the agenda for the meeting, and his initial set of slides is loaded into PowerPoint. He notices that a different Netscape window is prominently displayed in the montage, showing part of a review of last year's briefing that he studied carefully on Friday afternoon. As he works on the slides, he decides to e-mail the laboratory director to ask if he should include some preliminary data in the presentation to answer some of the criticisms in that review. As he sends the message, Charlie wonders when, if ever, he'll get a reply, as the busy director is not known for his timely responses. Charlie works on the slides for another hour and then sends a copy to the printer. Checking the printer queue, he finds that he is behind three large print jobs. Mentally reminding himself to get the printout later in the morning, he decides to shift gears and review his notes before a lunchtime meeting.

As he selects the project montage from his board, the briefing materials disappear from his desktop and the updated montage is now visible on the wall. His recent efforts at writing a project paper are now on his desktop as well as his notes from the design session last month. As he contemplates his notes, he notices that the face of the laboratory director is visible on the whiteboard, layered on top of the briefing notes. Ah, the director is likely in the coffee area. Charlie intercepts the director and gets the quick answer he needed. As he finishes reviewing the design notes, Charlie realizes that his lunchtime meeting will convene shortly.

Charlie quickly saves his notes and grabs his lunch. Out of the corner of his eye, he notices that the briefing montage has a printer icon overlaid on top of it. The printout! Charlie heads off to retrieve his printout before the meeting.

3.2.2 Related Work

This research leverages and extends efforts in many areas of HCI, especially the extensive past work on multiple-workspace window managers—especially Rooms (Henderson & Card, 1986)—and the use of projected displays in office settings—especially Flatland (Mynatt, Igarashi, Edwards & LaMarca, 1999, 2000). Kimura is also influenced by, and builds on, research in context-aware and ubiquitous computing, ambient displays, and activity monitoring.

3.2.2.1 Multiple-Workspace Window Managers

Kimura is one of the first systems to extend the notion of "task," as defined in Rooms and subsequent systems, to "activities" that include more than just the documents and application windows currently being used. One implication of this distinction is that Kimura portrays past actions, including completed tasks (e.g. working with a now closed application), as part of an activity. Additionally, Kimura moves the iconic representation of the activity off the desktop into the physical office (onto the augmented whiteboard). The montages used as iconic representations of the activities are designed to convey what was actually being done in the task, not just what windows are currently open. The montages are constructed from images of the most "important" windows, with different measures of importance being possible. Furthermore, Kimura collects additional information about the activities, such as the status of print jobs, e-mail and collaborators, and use this information when generating the montages to support peripheral awareness of the state of the activities as a whole.

Kimura places the activity icons (montages) onto the augmented whiteboard to support awareness of background tasks (see the following section for a more detailed discussion of the system's use of the augmented whiteboard). Many of the navigation and interface design issues in Rooms, and subsequent systems, were designed to overcome the fact that only the focal desktop is typically visible. By having representations of all activities continuously visible on a large, interactive surface, Kimura takes advantage of users spatial abilities to organize, monitor and navigate directly to a desired activity.

3.2.2.2 Interactive Wall-Sized Displays

This work is also influenced by research in augmented whiteboard interfaces, in particular Flatland (Mynatt et al., 1999; Mynatt et al., 2000), as it strove to support informal work practices within individual offices. Kimura is designed to complement Flatland's interface. Each of the system's montages is a segment that responds to gestures for moving and annotating the segment. More generally, the whiteboard interface is designed to support casual inspection and organizational activities.

This work extends previous efforts in whiteboard interfaces by directly connecting interaction on the whiteboard with interaction on the desktop. As an extension of traditional desktop computing, the whiteboard hosts montages that act as links back to previous activities. Additionally, the whiteboard serves as the display medium for background awareness cues.

There has been substantial research in augmented whiteboards for conference rooms, including Tivoli (Moran et al., 1996) and i-Land (Streitz et al., 1999). Some of the interaction techniques for large display surfaces, such as "throwing" in i-Land were adopted in the Kimura interface. Likewise, the advanced projector display techniques used in the Augmented Office research project (Raskar et al. 1998) could enable users to paint an interactive whiteboard on any surface in their office.

3.2.2.3 Other Related Work

There have been a large number of systems that attempt to capture information from live desktop systems for a variety of purposes, and while Kimura does not share the same goals as many of these systems, there are many engineering concerns common among them. Manufaktur and the Task Gallery, mentioned above, are the closest to Kimura's goal of capturing as much information about running applications as possible (Büscher et al., 1999; Robertson et al., 2000). Lumiere (Horvitz, Breese, Heckerman, Hovel & Rommelse, 1998) is closest to the current implementation of the Kimura system, which uses Windows system-level hooks to get access to all applications and user activity. Like Lumiere, Kimura uses the information to build a model of user activity, although the end applications of this model are far different.

As mentioned in the introduction, this research leverages the same human perceptual abilities that motivated previous work in ambient and peripheral displays (e.g., Ishii & Ullmer, 1997). Kimura's montages act as peripheral displays that present information about background tasks in a non-obtrusive manner. One novel aspect of this work is the construction of ambient displays from actual images of the user's work, in contrast to using only abstract or iconic imagery. Kimura's montage styles are reminiscent of the "piles" that conveyed the age and type of items in desktop folders (Mander, Salomon & Wong, 1992).

Kimura can also be viewed as a context-aware application (Voida et al., 2002); to function, Kimura relies on the continued deployment of a context sensing and aggregation infrastructure such as the Context Toolkit (Dey, Salber & Abowd, 2001). At the time Kimura was originally designed, there were no other context-aware applications commercially available or in published the research literature that combined a detailed model of the user's multi-tasking activity with external context.

3.2.3 Interaction Design

Multitasking is a complex, albeit common, human activity. Piles of paper around the periphery of a desk are a physical manifestation of multitasking, indicating a repeated practice of pulling materials into the center for focused work and then collapsing them back into a pile on the periphery when attention is turned elsewhere. Phrases such as "keeping tabs on things" and "juggling as fast as I can" hearken to the need to constantly monitor multiple activities.

Kimura's design is intended to support these common multitasking practices. Constantly available visual representations of background tasks afford many interactions that support multitasking. The representations are available for perusal, reminders, and large-scale organization and prioritization. Moreover, the content of the representations serves to remind users of past actions. Finally, new information about a background activity can be attached to these representations, leveraging peripheral awareness capabilities.

3.2.3.1 Basic interaction with the wall display

Kimura's wall display is designed to support two fundamental types of interaction. First, and most importantly, users may treat the wall display as a peripheral interface for keeping track of the existence of and changes in background activities. Second, users are empowered to directly manipulate the montage images, in conjunction with other whiteboard tasks, while standing at the wall display. Selecting a montage triggers a task switch. This operation can be performed from the desktop or from the wall display. The contents of the past activity disappear from the desktop and reappear as a montage on the wall display (Figure 3.3).

Figure 3.3 Overview of the peripheral wall display and the desktop monitor. Two of the montages include annotations (a red scribble and the blue text "Due Wed").

Simultaneously, the contents of the new task appear on the desktop. The montage for the current task is also displayed near the desktop monitors. This near-periphery display allows the user to remain aware of contextual cues, such as a past browsing activity, that are no longer part of the active desktop. Moreover, any additions to the montage, such as annotations (described below), are also available for perusal. Montages retain their position on the wall display so that a background task will return to its prior location unless the user explicitly rearranges the montages.

Montages can be manipulated in the obvious ways on the wall display: moved, deleted and so on. Simple gestures are associated with these common behaviors; montages are segments as in Flatland (Mynatt et al., 1999), and therefore react according to a specified behavior when gesturing on them and adjust their size to fit their contents. Currently, the behaviors connected to montages are *moving* when selected, and *annotating* when de-selected.

Annotating montages is an example of an interaction that is well suited for the wall display; using the dry pens of various colors provided with the SMARTBoards, the user may annotate montages with simple ink.

3.2.3.2 Visualizing tasks with montages

Montages are peripheral representations of a user's working contexts. As such, they should express the semantics of the working contexts and their relationships in a non-intrusive, albeit suggestive, way. Over the course of the research, I explored various visualizations of the information conveyed by montages (see Figure 3.4 through Figure 3.6). In all of the montage prototypes, images from the user's actions in the working context are manipulated to provide a quasi-summary of past activity.

At this point, Kimura's montage designs are based on an informal understanding of the key characteristics of a working context's history; namely characteristics such as primacy (what consumed most of the user's time), recency (what were the last actions of the user) and inter-relationships (what parts of the tasks are performed in close concert with each other) are highlighted. These designs combine literal representations of the working context (application snapshots) with various visualization techniques to convey its history at a glance.

The montages are also designed to reflect a sketchy look in order to suggest that the information on the wall displays is peripheral to the active working context of the user; montages are shown with sketchy backgrounds in soft colors using a separate color for each montage.

Some visualization techniques are common to all three montage designs. For example, recency is encoded as transparency so that the most recently used documents are the most opaque; Kimura's design uses five levels of transparency. Another example is the design's use of watermarks (highly translucent images). In many cases the low-resolution images of documents are not entirely readable; their visual utility is similar to a

thumbnail image. Therefore, to enhance the recognizability of the images, these montage designs incorporate watermarks of the application icon for major applications.

Figure 3.4, Figure 3.5, and Figure 3.6 demonstrate three major organization schemes for montages:

Figure 3.4 Two montages arranged in a spiral based on the decreasing significance of their contents.

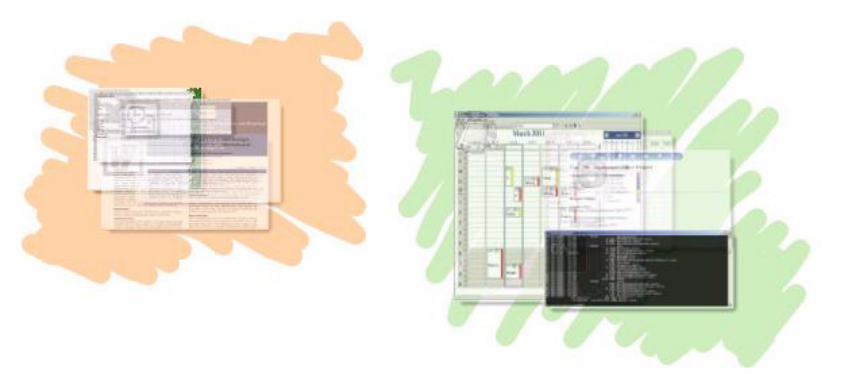


Figure 3.5 Visualization of two montages retaining original spatial layout of documents.

Figure 3.6 Two montage visualizations based on relative interdependence of documents.

3.2.3.2.1 Spirals of Significance

In the first design, documents are organized according to their overall significance in the task, as the most significant documents should be more easily recognized. As shown in Figure 3.4, document images are organized in a spiral with the most significant document placed in front and the less significant documents spiraling out in order of decreasing significance. The sizes of the documents also decrease according to their significance. The current significance rating is a measure of how much time was spent on a particular item, weighted by how recently it was used.

3.2.3.2.2 Preserving Spatial Relationships

Since the spatial organization of documents on the desktop is often visually salient for recall (Hutchins, 1995), an iconic rendering of this relationship may be easily recognizable by the user. As shown in Figure 3.5, document images in the montage are placed akin to where they were on the desktop display, and their sizes are also relatively the same. Additionally, the stacking order of the documents is preserved so that the most recently used document is represented at the front. Montages retain the same aspect ratio as the desktop display (0.75 in this case).

3.2.3.2.3 Relative Interdependence Mapping

Complex activities likely include a number of inter-related tasks; as different information sources are used for different purposes, sub-groups emerge in the working context. Likewise, documents may have strong similarity ratings due to common content. Exposing these relationships may help characterize the activity, especially activities that are left untouched for long periods of time.

The visualization in Figure 3.6 takes advantage of these relationships by using a modified version of the automatic layout algorithm presented by Szirmay-Kalos (1994). The measure of relative interdependence between two documents is currently based on the number of times the user has switched between documents.

The algorithm creates a graph of nodes and edges using a mechanical system simulation: nodes with edges tend to cluster together and nodes without edges repel one another. Edges may also be weighted; edges with stronger weights indicate that nodes connected by the edge will attract one another more strongly.

In the relative interdependence visualization, nodes are documents and there is an edge between two documents if the user has switched between the documents. The "connectedness" of two documents (the intensity of their edge) is calculated from the probability that the user will switch between the two documents. This measure is calculated using the actual switches a user has made between documents. In Figure 3.6, the top left document in the left montage has not been used much in connection with the other documents; in the right montage, the two leftmost documents have been frequently used together.

3.2.3.3 Background awareness cues

Montages also serve as anchors for background awareness cues related to a particular working context. Two examples are shown in Figure 3.2 based on the earlier scenario. When a person who is deemed critical to a background activity becomes

available, his or her face is shown on the montage. Kimura is implemented to look for e-mails sent to individuals where there has not been a reply. When one of these individuals is available in a public place, such as the coffee room, the montage is adjusted to note their availability. As faces are extremely salient visual cues, the Kimura montage design is intended to make use of them sparingly.

Another example is the use of tools that are left operating in the background. The status of these jobs, such as a print request, is reflected in the montage. Figure 3.2 also illustrates a completed print job for a particular activity.

3.2.3.4 Working contexts and the desktop

Other multi-desktop systems, such as Rooms, provide a variety of facilities for controlling which applications appear in the different desktops. The architecture of Windows 2000/XP, however, minimized the need for these facilities in Kimura. First of all, many of the small utility applications that were commonly shared across desktops are integrated into the taskbar, which is automatically shared across all desktops. Perhaps more importantly, if programs that use the old "multiple document interface" (where the application opens one large window and creates sub-windows for each document) are ignored, the applications themselves generally "do the right thing" when they (or their documents) are opened in the context of a multi-desktop manager.

When an application is running and the user tries to open it by clicking on its icon or one of its documents, applications that should only have one shared instance, such as messaging applications (e.g., Instant Messenger) or mail readers (e.g., Outlook Express), attempt to activate their window. Applications that use a one-window-per-document model (e.g., Word) activate the window for the already opened documents and create new windows for new documents. Some programs, such as web browsers, always create a new window when the user tries to open the application.

In Windows, multi-desktop managers function by using Win32 facilities to hide windows that are not on the current desktop. Since the Kimura desktop manager keeps track of the windows that are opened in each desktop, when a hidden window (i.e., one that was created on a different desktop) is activated, the desktop manager reveals it and adds it to the current working context. Therefore, it becomes part of the current desktop, and continues to exist in both desktops.

3.2.3.5 Inferring working contexts

The problem of inferring a person's working contexts is non-trivial. As a person goes about their daily activities, they interact with a multitude of documents (files, e-mail messages, and web pages) to accomplish each task. Kimura models a working context as a cluster of documents that relate to a particular task or activity. A basic problem that must be addressed, then, is how to tell which documents are associated with each working context. For example, when a new web page is accessed or a document opened, is it part of the current working context, the start of a new working context, or a signal to shift to some other existing working context?

At this stage of my research, I did not attempt to solve this problem. I explicitly elected to avoid automatic techniques because it is unclear how well they will work, and I did not want the success or failure of these automatic techniques to confound the study of the utility of peripheral information displays. Instead, Kimura enlists the help of the user by having them identify and explicitly switch between working contexts, using a set of lightweight tools to create, destroy, and manipulate working contexts over time.

3.2.4 System Architecture

Kimura's architecture can be broken down into five main components, as shown in Figure 3.7: desktop monitoring and management agents (for the Windows 2000-based

focal display), external context monitoring agents, tuplespace-based communication, activity interpretation agents, and the augmented whiteboard.

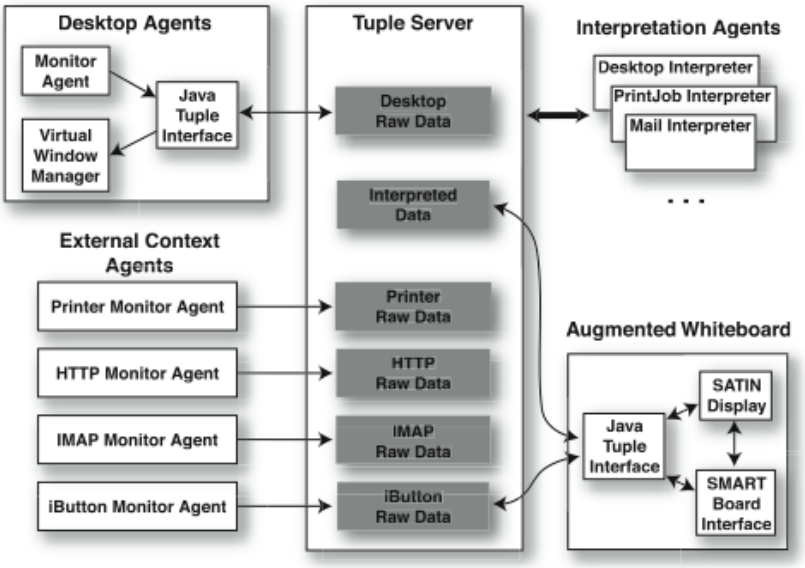


Figure 3.7 Architecture of Kimura. Arrows indicate primary data flow. The system is designed as a collection of agents communicating via shared tuple spaces.

In general terms, the desktop and external context monitoring agents continuously collect information about the user's activities and store it in the "raw" tuple spaces. The desktop agent also keeps track of which windows belong with which activities and switches which windows are visible when the user requests a different working context. The activity interpretation agents collect this information and use it to create a representation of the user's activities in the "interpreted" tuple space. The whiteboard process uses this representation to create the montages on the augmented whiteboard display and supports the interactions with the whiteboard.

3.2.4.1 Design Considerations

The architecture is designed to be:

- flexible enough for research exploration, and
- practical for real use.

To satisfy the first goal, the majority of the system (aside from some low-level Windows routines) is implemented in Java, and a blackboard architecture is used for communication. The system is designed as a collection of distributed agents communicating via centralized tuple spaces, implemented using the Java-based TSpaces architecture (Wyckoff, McLaughry, Lehman & Ford, 1998). This approach is well understood, and is also used in systems such as the Open Agent Architecture (Cohen, Cheyer, Wang & Baeg, 1994) and Interactive Mural (Fox, Johanson, Hanrahan & Winograd, 2000). Tuple spaces are robust in the face of process failure and allow each agent to be implemented independently.

To ensure that the system is practical for real use, Kimura is based on three design decisions:

- Kimura uses the low-level Windows hooks API to monitor and control applications. These hooks work with all applications, although they do not provide information in exactly the form required (e.g., it is hard to robustly acquire window images).
- Kimura does not change the behavior of the Windows desktop in any substantial way (aside from controlling which windows are visible). This approach contrasts sharply with the Task Gallery, for example, which replaced the Windows desktop with an entirely different metaphor.
- The desktop is controlled asynchronously, separate from the rest of the system. The cost of activity monitoring, data interpretation and display on the augmented whiteboard does not impact the performance of the desktop. Similarly, when the user switches activities, the desktop reacts immediately, regardless of the speed of change on the whiteboard.

In the remainder of this section, I describe the major components of the system, and close by discussing the engineering challenges of creating a system of this sort.

3.2.4.2 Desktop Monitoring and Management Agents

Kimura's focal display is a Windows 2000-based computer. Win32 "hooks" allow the system to intercept events, ranging from low-level input to window manager events, on all windows system-wide. A component running on the desktop system uses a DLL that provides callbacks to hooked events detected by the operating system. The callback information for each hooked event is packaged and sent as a Windows message to the *desktop monitoring and management agent* (written in Java), which stores the information in the *desktop raw data* tuple space.

This agent captures the entire history of the layout of windows on the desktop, and maintains a list of currently open windows for each activity. Each time a window is moved or resized or the window focus changes, the window layout is recorded. Each time a window acquires the focus, a snapshot of the window is taken and stored in a networked filesystem. This strategy ensures that the system has a relatively up-to-date image of each window, without overloading the system by capturing each window update. Since Windows only lets us capture the visible part of a window, capturing when the window has focus ensures that the window is not obscured.

The desktop agent also watches the tuple space for *SwitchMontage* tuples (see also the section below on the Tuplespace-based communication protocol), which signal that the user has switched to a different activity. When this tuple is seen, the windows for the current activity are hidden and those for the requested activity are exposed. The desktop agent also handles the exposure of hidden windows (from other activities) when they are activated, as discussed in the section on the interaction design.

3.2.4.3 External Context Monitoring Agents

In addition to monitoring a user's interaction with application windows, Kimura is also designed to incorporate relevant external information to provide a clearer picture of each activity. To illustrate the use of external context, the Kimura prototype monitors e-mail and web accesses, as well as the status of print jobs; all of the monitoring is currently done without instrumenting specific applications.

An HTTP proxy server (the *web monitoring agent*) monitors web access. The *printer* and *e-mail monitor agents* run on Unix mail and print servers. The e-mail monitor agent periodically scans a user's inbox and "sent mail" folders for new messages, correlates them based on message IDs, and writes a trail of mail activity into the *IMAP raw data* tuple space. The printer monitor agent watches the print queues for jobs created by the user, and writes status information to the *printer raw data* tuple space.

Kimura's design is intended to operate within a more general context system, such as the Context Toolkit (Dey et al., 2001) or CoolTown (Kindberg et al., 2002). For its prototype implementation, the system uses Dallas Semiconductor i-Buttons[1] to trigger the sorts of external context events that such a system would generate (such as the arrival of a colleague). The iButton events are written into the *iButton raw data* tuple space by the *iButton monitor agent*, and used by various agents for testing and demonstration purposes.

3.2.4.4 Tuplespace-based Communication

As mentioned above, the use of tuple spaces (and other blackboard systems) is common in current distributed interactive systems (e.g., Wyckoff et al., 1998), and offers a number of advantages over connection-oriented event-distribution schemes. These advantages include resistance to isolated process failure, global shared state, and the

[1] http://www.ibutton.com, accessed 15 January 2008

simplicity of using an unstructured data store. TSpaces also provides persistent tuple spaces, greatly simplifying the debugging of individual agents.

There are two situations that typically cause problems for tuple spaces. First, they have trouble dealing with high-volume updates that need to be distributed with low latency, making them inappropriate for distributing data such as mouse motion events. Second, the performance of the event matching algorithms suffers if a tuple space becomes large. Kimura addresses the first concern by not sending high frequency data over the network (i.e., the system does not capture mouse motion, but, rather, actions like button and keyboard presses). The system addresses the second concern by using multiple tuple spaces, as shown in Figure 3.7. The *raw data* tuple spaces (there are currently five) are used to store the transient data collected by the various monitors. The *interpreted data* tuple space contains the processed data that is used to create the montages.

3.2.4.4.1 Data Flow

The arrows in Figure 3.7 show the data flow in the system. Most data flows from the monitor agents, through the interpreter agents, into the interpreted tuple space, and finally into the augmented whiteboard process. The whiteboard process also monitors the iButton raw data space for simulated context tuples, which it uses when generating the montages.

Control data flows in the other direction, from the whiteboard process into the interpreted data space. The whiteboard stores both the montage annotations and *SwitchMontage* tuples (created when the user selects a montage to switch to) in the interpreted space. Any monitor or interpreter agent that cares about activity changes can subscribe to receive *SwitchMontage* tuples. For example, the desktop monitor agent switches the contents of the desktop to the windows for the specified activity when it receives a *SwitchMontage* tuple.

3.2.4.5 Context Interpretation

Java agents that collect data from the raw tuple spaces merge the data into internal activity timelines and store the information needed by the augmented whiteboard in the interpreted data space do the context interpretation.

The principal agent is the *desktop agent*, which extracts a representation of the current document activity from the desktop raw data space. I implemented two other agents as examples of the potentially useful activities. The *printer agent* extracts the status of print jobs in the current activity from the printer raw data space, and creates *printJob* tuples associated with the current montage. The *e-mail agent* extracts from the IMAP raw data space a list of e-mail messages that have been sent during the current activity, for which replies have not been received, and creates *unrepliedEmail* tuples associated with the current montage.

Even though the current collection of montages only uses a fraction of the data collected (e.g., the system uses only the last position, size and image of each window, and currently ignores the web access log), the architecture makes it simple to experiment with alternative montage styles and content. The interpreters maintain complete internal representations of the merged data and can access any of the tuple spaces, including the interpreted data space, as desired. Therefore, they can be modified relatively easily to extract the alternate collections of activity information and add it to the interpreted data store.

3.2.4.6 Augmented Whiteboard

The augmented whiteboard is implemented as a single process with three main components, all implemented in separate threads: graphical input/output based on SATIN (Hong & Landay, 2000), communication with the interpreted and iButton raw data spaces (described previously), and communication with multiple SMARTBoards.

The whiteboard display class is an instance of a SATIN Sheet: montages are implemented as segments on top of SATIN Patches, annotations are basic SATIN Strokes, and montage image elements are implemented using the SATIN image class. Kimura uses standard and custom SATIN interpreters and recognizers to control the montages.

On start up, the whiteboard reads tuples for existing montages from the interpreted data space and creates the initial display. The whiteboard process then subscribes to the interpreted data space for similar tuples and reflects any tuple space updates on the display. If any *unrepliedEmail* tuples exist for a montage, the process monitors the iButton space for the appearance and disappearance of the recipient of the e-mail, and uses this information as discussed in the interaction design.

The whiteboard process talks directly to the two SMARTBoards on the office wall. It translates tool position messages to the coordinate system of the SATIN window, based on the ID of the SMARTBoard (provided by the SMARTBoard API), and sends synthetic mouse events to the SATIN Sheet. Tool change messages (e.g., blue pen picked up) are also sent to the SATIN Sheet and used for actions such as coloring montage annotations.

3.2.5 Summary and Lessons Learned

Kimura is an important step toward unifying focal and peripheral office displays in a way that supports existing work practices. By adding activity montages to an augmented whiteboard, the system integrates peripheral awareness of background activities into the existing work area and allows users to organize and annotate these montages as they would other content on their whiteboard.

The design of the Kimura system is based on a focused, simple model of activity. Its design supplants the traditional “desktop,” application-and-document metaphor and allows users to manage their ongoing activities in the same way that they conceive of and

manage their tasks in the real world. It also builds upon the findings of previous studies of knowledge work, allowing users to organize their work spatially and without needing to explicitly name or label information in order to work with it. In addition, Kimura's persistent visualizations of ongoing activities and informal interaction style of the electronic ink surface closely correspond to existing knowledge work practices.

However, several design decisions were made to limit the scope, and therefore the complexity, of the design space for the Kimura project. For example, the system is designed for use in one worker's personal office, and primarily by that single user. Activities are represented as "flat" collections of documents, as opposed to hierarchical representations or representations with variable perspectives, so that the montage visualizations for each activity can be more readily evaluated. The system also provides only incidental support for collaboration in its notification cues of colleague availability; no explicit tools for collaboration or information sharing are included in the initial system design.

Several technical obstacles revealed during the implementation of the Kimura system also serve as lessons for the future development of activity-based systems. Aside from the usual challenges associated with building any complex distributed application, such as communicating huge amounts of image data between components and dealing with replicated distributed state (MacIntyre & Feiner, 1996), the most significant engineering challenges were related to monitoring and controlling the activity on the Windows desktop. While the Hooks API allows the system to monitor window, mouse and keyboard activity, it does not provide any semantic information about what is happening within the applications, such as what files are open and what windows are associated with an application. As noted by Robertson et al. (2000), without a standard application interface to inspect the applications, activity-based systems must resort to dealing with applications on a case-by-base basis. It is even difficult, for example, to sort out splash screens and dialog boxes from content windows.

Another shortcoming of Windows that Kimura exposes is an inability to restore an application's state when the system is restarted or the desired document or application window has been closed. Assuming that the system could be designed to retrieve more than basic window handles and thumbnail images for a selected set of applications (e.g., the URLs of web pages accessed, the folders and message IDs of e-mail messages sent and received, and details about office documents edited), there are still a large set of problems that must be dealt with in order to properly reopen a document. Ensuring that the window is in the correct position on screen, scrolling the document to the correct location, and reinstating the previous interaction mode (e.g., selection region, active tool, undo/redo history) is not possible in the general case.

3.3 The Sharing Palette

Initial activity-based systems reported in the literature generally lacked sophisticated support for collaboration. However, most descriptions of knowledge work acknowledge information sharing as a critical component of collaboration for knowledge workers.

In research focused on user interface support for file sharing, I explored users' current practices and needs around file sharing (Voida, Edwards, Newman, Grinter & Ducheneaut, 2006). The survey- and interview-based study focused on the workplace file-sharing practices of ten employees at a medium-sized research corporation. My participants reported that they shared files with an average of 7 individuals or groups on a regular basis; that they shared a wide variety of file types, ranging from business documents and paper drafts to music, ideas, and TV shows; and that they accomplished their sharing using relatively few sharing mechanisms. Of these, e-mail was by far the most prevalent, used for 43% of all reported instances of file sharing. In 13% of the responses, participants reported using multiple mechanisms (e.g., e-mail and a shared network folder) simultaneously to share a particular type of file with a recipient or group.

Finally, the participants reported experiencing three main classes of breakdowns over the course of their file sharing experiences:

- forgetting what files had been shared and with whom,
- difficulties in selecting a sharing mechanism with desired features that was also available to all sharing participants, and
- problems in knowing when new content was made available (or updated) when acting as a sharing recipient.

In order to determine the most useful set of user interface affordances to explore, I looked to existing sharing tools and the attributes of those tools most influential in the selection of a sharing mechanism. Through an analysis of the sharing tools most commonly used by the study participants, I identified the affordances of each and enumerated a set of user-visible dimensions along which these tools varied. The attribute of utmost importance was the scope of sharing enabled by the tool; the sharing tools most commonly available were those used most often for sharing files. Since addressing is directly related to with whom information should be shared, it can also likely be described as one of these primary characteristics. The study participants reported the most breakdowns with respect to the visibility and availability of notifications provided by a particular sharing tool. While these affordances seem to be at a much finer level of granularity than scope and addressing, because of their impact on the overall sharing interaction design, they can also be considered critical attributes. At a higher level, the orientation of a sharing mechanism with respect to whether information is being *pushed* to a recipient or simply made available and *pulled* from the sender when convenient appears to be critical in defining the interaction nature of that mechanism; with any of the other categories, altering the mechanism's characteristics within that category would still preserve the effective character of that sharing mechanism.

Based on this analysis, I implemented an interface called a *sharing palette*, which provides a platform for exploration and experimentation with new modalities of sharing

(Figure 3.8). The sharing palette features a hybrid sharing modality incorporating some aspects of push-oriented sharing and some aspects of pull-oriented sharing. It also features flexible addressing through the ability to share files publicly, with individual recipients or with ad hoc, semantically meaningful groups of recipients. The palette interface provides persistent visibility of shared files and potential recipients, and it includes a variety of notification features, which are designed to promote awareness of changes to the sharing state.

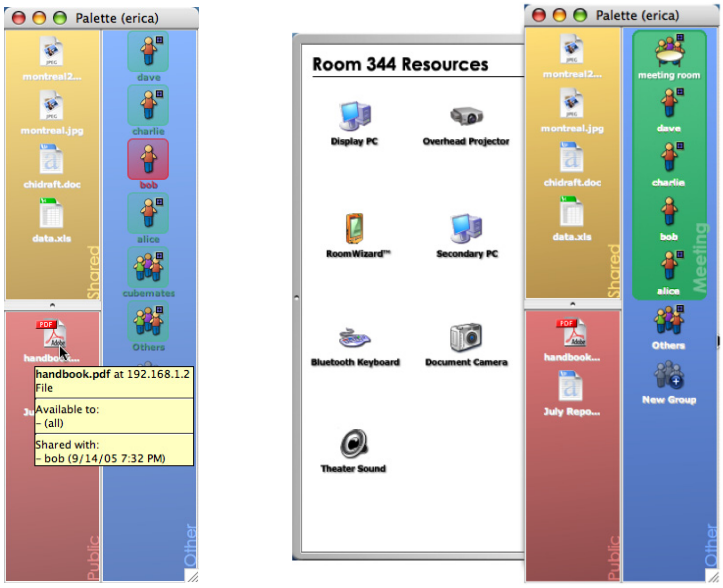


Figure 3.8 Prototypes of the sharing palette user interface. On the left, using the palette to discover with whom the file *handbook.pdf* is currently shared. On the right, an initial (non-interactive) prototype of an activity-aware palette, providing sharing services and control over peripherals during a meeting.

While not directly related to supporting activities, the interface reflects reported knowledge work practices (as they relate specifically to file sharing) and is designed to be extensible in ways that would enable its integration into activity-based desktop systems. The lessons learned about the importance of flexible addressing, persistent visibility of information, and appropriate notifications reflect observations about knowledge work in

general, and will also likely be applicable to user interfaces designed to support knowledge work and activity-awareness.

CHAPTER 4

CHALLENGES IN REALIZING ACTIVITY-BASED COMPUTING SYSTEMS

Based on the reported knowledge work practices, cognitive and psychological theories of human activity, and technical explorations outlined in the previous sections, I have identified three primary challenges for representing and supporting activity: activities are a part of a fluid work practice, activities encapsulate evolving work, and activities are collaborative (adapted from Voida, Mynatt & MacIntyre, 2007). They are challenges (as opposed to simple design guidelines) because they are currently not well reflected in mainstream computer systems and doing so will require overcoming one or more technical obstacles or users' established expectations about the user interface paradigm at play.

In this section, I unpack each of these three challenges individually. I provide specific suggestions for how interfaces might support activity-based knowledge work in light of theory and practice and explain the barriers that currently exist for building systems that provide this type of support. (Table 4.1 provides an overview of this section, including the challenges, their basis and the related barriers.)

4.1 Challenge 1: Activities are Part of a Fluid Work Practice

Activities represent individual, distinguishable components of a larger, fluidly interconnected knowledge work practice. Many studies of knowledge workers emphasize the fractured and frequently-interrupted nature of the work environment; in order to provide support in these situations, activity-based computing systems must echo the agility of the human user in switching between and creating new activities. These systems must also play an active role in reminding the user of ongoing activities and provide

Table 4.1 Challenges in realizing activity-based systems.

	Challenge	Basis in Observations and Theory	Technical or Interface Inertia-Related Barrier
1	**Activities are part of a fluid work practice**		
1.1	Activity management must support multitasking and interruptions	Observed prevalence of multitasking and interruptions	Existing window system operations are almost exclusively application-oriented; window group operations available only with additional tools
1.2	Activity management must remind the user of ongoing activities	Observed use of activity-management artifacts and GUI elements as reminders; observed problems with forgetting in sharing study	Current overview visualizations (e.g., Windows' TaskBar and OS X Dock) are quite impoverished; tension between minimizing screen real estate and activity visibility
1.3	Activity management must minimize additional "metawork"	Metawork already accounts for almost 10% of daily work	Maintaining representations without additional user interaction requires hard-to-access information about state of applications, window system
2	**Activities encapsulate evolving work**		
2.1	Activity representations must incorporate diverse kinds of information	Observed activities differ based on their motives, contacts, resources & tools	Difficult to aggregate relevant information from application- and filesystem-based silos
2.2	Activity representations must support evolutionary information classification	Observation that knowledge workers are informed while transforming information; "piles" and "files" organizational practice	Saving files typically requires known hierarchy and name before information can be stored; re-structuring previously filed material is difficult and onerous
3	**Activities are collaborative**		
3.1	Activity representations must reflect communicative aspects of collaboration	Observed importance of communication (e-mail and face-to-face meetings); importance of social context in activity theory models	E-mail typically exists in a silo independently from other stored information; storage hierarchy does not reflect history of collaborative interactions
3.2	Activity sharing must prevent unintended information disclosure	Observed behaviors of hiding sensitive information; reported problems remembering which information is shared and with whom; findings of sharing and privacy preferences study	Specifying access control is typically complex in order to provide desired flexibility; maintaining visibility of shared information and sharing scope difficult due to limited screen real-estate
3.3	Activity sharing must accommodate differences in granularity of activity specifications	Observation of wide individual differences in activity specification granularities; activity theory explanation of activities at different levels of analysis	Differences will be largely semantic and difficult to resolve without substantial user intervention

appropriate support without requiring excessive additional "metawork" actions on the user's behalf.

4.1.1 Challenge 1.1: Activity Management Must Support Multitasking and Interruptions

Various studies have reported on the average number of activities a knowledge worker juggles each workday, ranging from 10 (Czerwinski, Horvitz & Wilhite, 2004; González & Mark, 2004) to 11.5 (Mark, González & Harris, 2005) in recent studies, and up to 58 short activities in an older study of managers' practices (Sproull, 1984) [1]. The overall length of time spent working on each activity has been reported as an average of 53 minutes (Czerwinski et al., 2004), with an average of 11 minutes 4 seconds spent at one time on an activity before switching to another activity or being interrupted (Mark et al., 2005). One study observed that knowledge workers are interrupted an average of four times per hour (Czerwinski et al., 2004). At times, interruptions cause only a temporary shift away from an activity, with work resuming where it was left off when the interruption is resolved. However, this has been shown to happen only around 60% of the time (O'Conaill and Frohlich, 1995; Czerwinski et al., 2004).

The current application-centric model of desktop computing typically maps windows on the screen to individual documents. While this approach allows users to share screen real estate among several open documents and to quickly and fluidly move among them, the fact that activities often consist of documents represented in multiple windows necessitates that users either move or close and open several windows to accomplish an activity "context switch." Given that this has been reported to occur on average once every 10 minutes, this represents a significant expenditure of effort on the user's part.

[1] While this finding differs from others by a factor of five, it is unclear whether Sproull's activity count represents unique activities or uninterrupted blocks of working time during the day. I presume that the correct case may be the latter, based on the fact that Sproull's reported time spent per activity quite closely matches the average time spent on an activity before being interrupted by other researchers.

In my own study of activity and window management (see section 2.2), I saw a correlation between window manipulation frequency and activity switches; users would generally open and position the windows they needed as they were initiating or resuming an activity. The Kimura system provides one possible solution to this problem: using a "focus+context" approach (after Furnas, 1986) based on the use of a virtual desktop on the primary computer display and an overview visualization on a secondary, peripheral display—in this case, a front-projected electronic whiteboard. This approach provides a lightweight solution to both issues, as it is easy for the user to manage groups of windows through the tools provided by the virtual desktop manager and also to maintain a sense of task awareness through the persistent visualization of background activities. However, the virtual desktop metaphor begins to break down when applications are shared across multiple virtual desktops—most notably e-mail, IM, and calendar applications.

In order to support multitasking behaviors, activity-based systems need to provide these kinds of window management tools that apply across groups of windows, rather than just operating on a window-by-window basis or across all windows globally (e.g., Apple's Exposé functionality in OS X). This coverage of the middle ground is important since it is based on observations that, in practice, users often rely on multiple windows to accomplish a single activity and that switching between activities is a very common task. There are also some interesting opportunities to link group-level window manipulation operations with potential sources of interruption or task switching, such as the use of e-mail or knowledge about a user's upcoming calendar events. It might be useful, for example, to make windows related to a background activity available for quick access when the user opens and reads an e-mail written by a contact also associated with that activity.

4.1.2 Challenge 1.2: Activity Management Must Remind the User of Ongoing Activities

> People organize their desks in part so that they can find things…[however] an equally important function of most desk organizations is *reminding*.... The distinction between finding and reminding rests on intentionality. If you become aware of something you intended to find, then the finding function has been served. But if, in the course of doing one thing, you become aware of something else without intending to, you have been reminded of the second thing (Malone, 1983).

Malone's observations support the claim that a primary function of organizing work into activities is to help remind the worker about the current state of the work. Other research has revealed that one strategy for maintaining activity awareness involves the maintenance of artifacts that represent various ongoing activities; these artifacts "always appear in a visible spot of the working space so they can be consulted constantly" (González & Mark, 2004). This behavior manifests itself in a variety of ways. 72% of one study's respondents reported having sent reminders to themselves via e-mail; 83% reported leaving messages in their inbox as reminders (Ducheneaut & Bellotti, 2001). Knowledge workers have been found to use information displayed in windows as reminders to switch tasks (Hutchings & Stasko, 2004). Miyata and Norman's theory of activity, based on empirical research in cognitive psychology, states that "reminding is required if suspended activities are to be resumed at the appropriate time or place" (Miyata & Norman, 1986).

Virtual desktop managers, the most commonly-available class of activity-based systems, are excellent tools for assisting users in partitioning their multitasked activities and minimizing the clutter of their screens, but they generally do a very poor job of providing awareness about activities that have been relegated to the background. There is an inherent tension between minimizing the use of screen real estate used by irrelevant applications, documents, and task-management tools and providing visibility of all activities.

The Kimura system provides a persistent display of suspended activities as abstract montages on one or more peripheral electronic whiteboards, a location not typically used for focused, individual work. The montage representations are designed to provide a salient, at-a-glance overview of the most significant components of an activity, with several visualization options intended to assist the user in resuming the task after a wide range of interruption lengths. The montage visualizations also incorporate notification cues to provide reminders about pending actions within activities, such as print jobs that may need to be retrieved or colleagues from which e-mail responses are outstanding.

Perhaps the most critical aspect of reminding is the visibility of the reminder. Activity-based systems will need to take advantage of strategies like "focus+context" visualizations (after Furnas, 1986) or notification aggregation mechanisms (such as Growl on the Macintosh platform[2]) in order to utilize screen real estate efficiently but provide appropriate reminders of the current activity state and notifications of any changes to it.

4.1.3 Challenge 1.3: Activity Management Must Minimize Additional "Metawork"

With all the interruptions and self-imposed activity switches inherent in everyday work, knowledge workers have developed practices for maintaining an overall sense of task awareness:

> Individuals spend part of their day on a set of activities that is not connected with any specific working sphere but rather related to the management of all of them. We call these activities *metawork*. People periodically conduct metawork throughout the day, which involves coordination, checking activities, organizing email, organizing their desk at the start or end of a working day, and catching up with teammates on what they have missed. People spend an average of 44 1/2 min. per day conducting metawork, and similar to working spheres, this work is also conducted in shorter chunks of about six and a half minutes at any one time (González & Mark, 2004).

[2] Growl client software and SDK is available at http://growl.info, accessed 15 January 2008

In their later study, Mark et al. (2005) noted that metawork comprised about 8% of all work accomplished per day and that it was an activity that was rarely interrupted. Reported metawork practices include creating special inbox folders in the e-mail client, keeping printouts of e-mail messages and meeting notices, use of traditional activity-management tools (e.g., planners), and post-it notes (González & Mark, 2004). However, even with these organizational workarounds firmly entrenched, there is apparently room for improvement: "The design ideas most frequently offered by the participants revolved around…reminding, including the potential value of cross-application project and to-do list tracking" (Czerwinski et al., 2004).

It is possible to infer with some accuracy when the user is transitioning between activities based on their patterns of interaction with windows on the screen (Nair, Voida & Mynatt, 2005; Stumpf et al., 2005). However, these methods do suffer from misidentification of activities, which requires users to provide appropriate training feedback for the underlying AI system. There are also technical challenges in identifying what resources are used as information sources for a given activity; while it is generally easy to passively monitor the filesystem to determine what files are changed over the course of an activity, few operating systems provide robust hook mechanisms for consistently determining when files and network pipes are being read. Based on the current state of application scripting languages and operating system API support, most activity-based systems will require some degree of input and/or feedback from the user to capture relevant information about the state of and motivation behind a given activity.

The Kimura system's use of virtual desktops to implicitly distinguish among ongoing activities is a good example of how activity-based support can take advantage of existing work practices and provide sophisticated capabilities not currently offered, but with little additional interaction "cost" on the user's behalf. Likewise, the sharing palette provides lightweight features for creating and modifying sharing distribution groups and aggregating low-value notification cues to reduce distractions. The application of these

types of lightweight interaction mechanisms in activity-based systems can serve to reduce potential resistance to adoption due to an imbalance in the perceived cost and benefit of using the technology.

4.2 Challenge 2: Activities Encapsulate Evolving Work

Boer, van Baalen and Kumar's revised model of activity as comprised of a goal, mediating tools, and a social context but also changing over time and affecting the structure and perception of other activities reflects the basis (directly or indirectly) for much of the contemporary research into activity-based systems (2002). The ideas that activities encapsulate various artifacts and relationships in knowledge work and that they change over time, both in actual composition and in how they are perceived and understood by their participants, constitute substantial challenges in the development of systems to support activity-based knowledge work.

4.2.1 Challenge 2.1: Activity Representations Must Incorporate Diverse Kinds of Information

Observed knowledge work is comprised of activities distinguishable based on their "whole web of motives, people, resources, and tools," with tools including "documents, reference materials, software, or hardware" (González & Mark, 2004). Activity theory describes activities primarily in terms of their objects (objectives), but also recognizes the diversity of tools used to accomplish each activity and the social context in which the activity occurs (Engeström, 1987). However, situated action warns against representing and reflecting activity as rigid and inflexible, due to the spontaneous, ad hoc nature of "real-world" work (Suchman, 1987).

However, as noted above in Challenge 1.1, current window management systems generally provide only document- or application-level management tools; facilities for managing groups of document or application windows together are relatively

impoverished. In addition, it has been noted that current tools require users to maintain their own organizational hierarchies for their information, often in several locations simultaneously (Bergman, Beyth-Marom & Nachmias, 2006). This "project fragmentation" problem is symptomatic of a major technical barrier to realizing activity-based computing systems: mainstream operating systems typically leave almost all details of information organization up to the end user. With few exceptions, application "open" and "save" operations simply provide users a dialog box pointing to a common, catchall folder for storing user documents. Those applications that do maintain their own searchable database of resources (e.g., e-mail clients, some music and photo library software) typically do so for only a single type of file. Organizing these "silos" of information (as described by Bergman et al., 2006) is often left entirely up to the user, since it is typically difficult or impossible for applications to assist in organizing information across multiple silos.

My experience with the Kimura system demonstrates that representing activity as clusters of applications, documents, contacts and contextual information shows potential for fostering task awareness and the support of multitasking. The Kimura system allows users to organize and manage their work at the level of activities, as opposed to manually manipulating applications and documents. The system design is intended to lower the overhead of activity switching by allowing the user to switch easily between relevant groups of applications and documents as needed—much the same motivation as in systems like Rooms (Henderson & Card, 1986), Task Gallery (Robertson et al., 2000), and GroupBar (Smith et al., 2003). Kimura associates each activity with an individual virtual desktop on the primary desktop computer; the number and contents of a user's virtual desktops are used to identify the user's current activities and associate applications, documents, and external resources with those activities.

In order to function as useful tools for knowledge workers, activity-based systems will have to present a workable solution to the information organization problem. One of

the most tangible benefits such a system can provide will be in relieving the user from having to manually manage multiple silos of data; using some notions of activity as the organization structure for these silos would both echo existing real-world work practices and allow information organization to occur concurrently with the user's management of windows and screen real estate (see Challenge 1.1, above). Activity-based computing systems must also maintain a balance between flexibility and complexity in their representations and organizational schemes in order for applications to be able to utilize the modeled data and for users to be able to manage their representations without the models becoming overly brittle.

4.2.2 Challenge 2.2: Activity Representations Must Support Evolutionary Information Classification

User studies and intuition both suggest that the activities that a knowledge worker engages in change—sometimes dramatically—over time. While representing and reflecting these changes is necessarily an important function of activity-based systems, a more difficult observation that must be taken into account is that the way that users *think* about information and activities evolves over time as the information and activities inform them (Kidd, 1994). Recent extensions to activity theory posit a fluid progression between actions and operations, echoing this notion that how users conceptualize activities—whether they require attention or become routine and how they relate to other activities they are involved in or have experienced—changes over time (Boer et al., 2002). Finally, the act of labeling information and activities is informed, in part, by a knowledge worker's collaborations, and the labels become part of the work product when it is handed off to others.

Mainstream operating systems and applications place a heavy emphasis on *naming* and *filing* information, so much so that the specification of a name and relevant location for a document is typically a prerequisite for storing it to disk. While this focus

on a discrete name is perhaps appropriate for a deterministic computer system, it is widely recognized that this may not be the best case from a user's perspective, particularly in certain circumstances:

> Naming a file and deciding on its location is a common, albeit heavyweight task, too heavyweight for informal interaction with a whiteboard. Simply determining a name for content that is loosely associated with any product or deliverable is difficult (Mynatt, Igarashi, Edwards & LaMarca, 1999).

Malone (1983) makes an even more general observation, based on his observations of how knowledge workers manage their physical workspaces: "The difficulty of deciding how to classify something can be an important barrier to filing the information." While naming and filing do have value in the long run, particularly for finding and retrieving previously filed information, the emphasis that most computer systems place on these tasks is somewhat out of synch with real-world work practices. Furthermore, the fact that naming and filing are both generally given to be fully user-specified, if a knowledge worker changes the way that they think about information once it has been stored, re-structuring this previously organized material can be quite difficult and time-consuming.

The Kimura system uses one or more electronic whiteboards to provide peripheral displays of background activities. These whiteboards extend existing electronic whiteboard interaction techniques (see Igarashi, Edwards, LaMarca & Mynatt, 2000; Mynatt et al., 1999; Mynatt, Igarashi, Edwards & LaMarca, 2000; Hong & Landay, 2000). Kimura's montages were implemented as special cases of Flatland-style segments and, therefore, inherit some of Flatland's desirable properties for informal activity organization: "By default, each segment in Flatland is automatically saved…without requiring an explicit action or input from the user" (Mynatt et al., 1999). While the informal interaction style of Kimura's electronic whiteboards closely corresponds to knowledge work practices related to spatial and evolutionary organization of information, Kimura does not provide user interface mechanisms for naming or filing activities once

the user has been "informed" by the information (Kidd, 1994). The system also does not implement user interface tools for re-organizing existing activities, limiting the practicality of the system.

The idea that naming is a byproduct of knowledge work suggests that activity-based systems must allow knowledge workers to work with unlabeled activities and add and refine labels classifying them over time. One approach to instantiating this in the user interface would be to utilize a metaphor that reflects the distinction Malone (1983) draws between storing objects in "piles" for short-term, less-structured storage versus storing them in "files" for the long-term (for example, see Mander, Salomon and Wong, 1992). There may also be value in providing explicit representations of activities as reflecting their evolution over time (for example, see Rekimoto, 1999).

4.3 Challenge 3: Activities are Collaborative

Most knowledge work is inherently collaborative. Even if collaboration isn't an integral part of a given activity, the activity almost certainly draws upon information that was created by others at some earlier point in time or results in some deliverable that is then handed off to others (Tang et al., 2007). Recognizing the *mediating* role of the digital work environment in enabling users to collaborate meaningfully is a critical step to ensuring the success of these systems.

However, as the large, diverse body of literature in the computer-supported collaborative work (CSCW) community suggests, supporting effective collaboration is rarely a trivial undertaking. Technical issues involving the exchange of information, preservation of state, and graceful operation in the face of network failures, coupled with social issues regarding awareness, negotiation about the roles that collaborators will play, and privacy—to name just a few—abound. The three most significant challenges in supporting collaboration with respect to activity-based computing are in situating work appropriately within the context of communication and information sharing, preventing

the unintended disclosure of information, and accommodating differences among the ways in which collaborators establish boundaries around their activities and groups of work artifacts.

4.3.1 Challenge 3.1: Activity Representations Must Reflect Communicative Aspects of Collaboration

Practically speaking, communication and communication-oriented collaboration constitutes a substantial percentage of knowledge work accomplished throughout the day. González and Mark (2004) report that an average of 18.9% of the workday is spent in unscheduled meetings; the participants in Czerwinski, Horvitz and Wilhite's diary study (2004) report that 23% of their tasks during the day "could best be described as 'email.'" Ducheneaut and Bellotti (2001) describe how commonplace it is for knowledge workers to exchange information in preparation for a face-to-face meeting using a computer-mediated communication medium: "Document exchange is generally linked to meetings—both before and after them. Many of our respondents send agendas through e-mail (65 percent) or actions (69 percent) through e-mail." From a theoretical perspective, the activity theory representations also place a similarly heavy emphasis on collaboration and communication in activity, particularly in the importance of utilizing a division of labor to accomplish complex or difficult tasks (Vygotsky & Cole, 1978; Engeström, 1987).

Extending the technical obstacle described in Challenge 2.1, the multifaceted nature of activity makes it difficult for applications to provide cross-cutting support for representing and managing activity across multiple types of information, histories of communication, and instances of collaboration. For example, while it is possible to perform a search within an e-mail client to determine when a particular document was shared and with whom, this information is not normally reflected along with the document in the filesystem hierarchy. Because the filesystems included with most

mainstream operating systems are primarily geared toward supporting individual work, they lack the additional context and rudimentary version control features that are inherently available when files are exchanged via e-mail[3]. Similarly, colleague contact information is also generally stored in a separate "silo" and managed out of context with documents in the filesystem with which they might otherwise be associated.

There are several design considerations that would enable more robust collaboration support for activity-based knowledge work. First and foremost, other individuals and groups must be represented as first-class objects in computational models of activity. One potentially useful way to incorporate colleagues into activity representations is to leverage and visualize the relationships between ongoing activities and naturally occurring virtual and real-world social networks (e.g., Nardi, Whittaker & Schwarz, 2002). Second, it would be beneficial to integrate instances of communication and collaboration as milestones in the representation of activities. The importance of face-to-face meetings in knowledge work and the fact that so much information exchange occurs around meetings and through e-mail communication threads suggest that the meta-information about when collaboration occurred and with whom collaboration took place might be as useful an index to a document as a filename or the folder hierarchy in which a file is stored.

4.3.2 Challenge 3.2: Activity Sharing Must Prevent Unintended Information Disclosure

As more detail about a user's actions and the context surrounding his or her work are captured and stored, the risk of having this potentially personal information inadvertently shared with others during collaboration grows. Knowledge workers have been observed intentionally hiding windows containing sensitive or personal data when

[3] Incidentally, this might help to explain why so many activity-based computing systems described in the literature use the e-mail client as the locus of activity management, (e.g., Gwizdka, 2002; Bellotti, Ducheneaut, Howard & Smith, 2003; Kaptelinin, 2003; Geyer, Vogel, Cheng & Muller, 2003).

not directly accessing the window's contents (Hutchings & Stasko, 2004). A recent study on users' preferences for sharing and privacy reported that users vary substantially in their willingness to share, and as a result, one-size-fits-all permissions structures for sharing are inappropriate (Olson, Grudin & Horvitz, 2005).

Substantial systems-level work has been published in the CSCW community on the implementation of access control schemes that provide flexibility and descriptive power for maintaining the desired visibility and read-write permissions on individual shared objects (e.g., Shen & Dewan, 1992; Sikkel, 1997). However, these schemes still retain a degree of complexity from an end-user perspective, since the user still needs to assign roles to collaborators or keep access control lists up to date to enable intended sharing and prevent unintended sharing. Additionally, these schemes may not be appropriate for aggregate clusters of resources, should the user desire to "share" an entire activity with one or more collaborators. Finally, because the desktop metaphor was developed primarily to support individual computer use, visualizations of sharing privileges have traditionally not been persistently visible on items displayed on the desktop and in views on filesystem folders.

In my user study informing the design of the sharing palette prototype, one of the most-reported breakdowns in file sharing related to users forgetting which files they had shared and with whom. While most of the respondents reported that they agreed with the statement, "I am generally aware of all of the files that I am sharing, and with whom I am sharing them" (average of 3.7 on a 5 point Likert scale with 5 representing strong agreement), most of the sharing problems reported were related to respondents' discovering that they had forgotten which files they had shared, with whom they had shared them, or that they had difficulty managing the file permissions—either having set them incorrectly or having forgotten to change them at all.

Finding a balance between preventing unintentional information disclosure and fostering activity-aware collaboration requires difficult design trade-offs. Moreover,

systems must be designed with the social context of the workplace in mind; providing support for collaboration requires somewhat more subtlety than simply exposing all participants' activity representations and constituent resources to one another. Participants may wish to exercise varying degrees of control over how and when their resources and work processes are shared with their colleagues. They may also wish to provide a detailed specification of how their availability is shared with different colleagues. Finally, the organizational structure of the workplace may cause each collaborator to play different roles in the activity; as a result, each may need access to different activity representations or meta-information about the activity and contributions of its participants. Of some potential benefit are two findings from the privacy preferences study conducted by Olson et al. (2005): (1) generally speaking, users treat certain kinds of information similarly when assessing whether or not to share it with others (example classes include "work email and telephone number," "pregnancy, health information, and personal preferences," and "email content, credit card numbers, a transgression"); and (2), generally speaking, users treat certain kinds of individuals similarly when assessing whether or not to share information with them (example classes include "spouse," "manager, trusted co-worker," and "the public, competitors"). The idea of using classes of information and collaborators as a means for determining whether to share information and at what level of detail may be a useful starting point in developing tools to support this kind of collaboration.

4.3.3 Challenge 3.3: Activity Sharing Must Accommodate Differences in Granularity of Activity Specifications

At any given point in time, a single user may report being involved in several different activities, each specified at a slightly different level of granularity. For example, a user might be in the midst of writing a conference paper review, compiling a list of references for a proposal submission, and working toward a promotion. In a previous

study (Nair et al., 2005), I observed that there exist large individual differences in the granularity at which users define their activities. Kaptelinin, Nardi & Macaulay's *activity checklist* (1999) provides an explanation of how an *activity* at one level of analysis can be decomposed into *actions*—consciously carried-out subcomponents of an activity—and further into *operations*—routinized tasks that together constitute an action. Depending on their level of engagement with a task or their experience accomplishing it, knowledge workers might perceive of the same task at any of these three levels of detail. This holds true for individual users, as in the example provided above, but is even more pronounced when a single activity is viewed from multiple participants' perspectives. For example, a manager and a principal investigator might both be involved in the activity of completing a research project, but their perceptions of the importance of the activity, the tools, the actors involved, and specific goals might be quite different.

Three primary barriers challenge the realization of systems that handle activities specified at different levels of granularity properly. First, specification of activities is largely a matter of semantics; as a result, it will be difficult or impossible for systems to provide substantial assistance in identifying or labeling activities. Second, users will likely use different levels of granularity and kinds of labels to define and describe activities and resources for their own reference than they would when defining and describing activities and resources for consumption or reference by groups. Finally, even with all this observed experience in managing activities and multitasking, in a diary study of multitasking practices, participants "tended to use generic terms to describe their tasks…instead of using more specific, meaningful keywords to describe their activities" (Czerwinski et al., 2004). Finally, during collaboration, there will be cases in which two users need to coordinate between activities that each has established independently. The way that activities are modeled will determine the complexity of "merging" the models together, particularly for cases in which users conceive of and manage their activities at different levels of granularity.

The Kimura system represents activities based on the contents of a single virtual desktop on a primary desktop computer, placing few limitations on the contents or lifespan of a tracked activity. Kimura's montage visualizations are also designed to apply across activities specified at different levels of granularity. The visualization algorithm simply displays a combination of the longest-lived and most recently used window thumbnails associated with each activity. Regardless of how long- or short-lived the activity or how large or small the unit of work that it represents, the documents most salient to the user's experiences with the activity are included in the activity's montage visualization on the whiteboard.

Of course, supporting activities shared among two or more users complicates the situation. Suppose one user manages her tasks at a high, project-oriented level, for example, *annual project review* and *teaching*, and another user participating in the same activities manages his tasks at a much finer granularity, for example, *project review demonstration debugging* and *preparing computer graphics guest lecture*. This scenario is particularly likely when colleagues with different roles (such as a team member and a manager) collaborate on a single activity. Although it would be relatively straightforward to provide activity-level support for either of these users on their own, maintaining a shared representation of each of the users' collaborative activities at their preferred granularity, providing each user with appropriate views of the activities, generating notifications to each user for relevant changes in the activities, and coordinating changes in the structure of the activities over time become very complex.

CHAPTER 5

RESPONDING TO THE CHALLENGES: THE GIORNATA SYSTEM

Giornata is a set of activity-based extensions to the desktop user interface that provide activity-based resource storage, support activity-structured collaboration, use an extensible, evolvable tagging system to both name activities and access artifacts associated with those activities, and support both implicit and explicit interactions. This system integrates directly into the Mac OS X desktop and is compatible with current OS X applications. An overview of the Giornata user interface is illustrated in Figure 5.1, with callouts annotating some of the system's significant features.

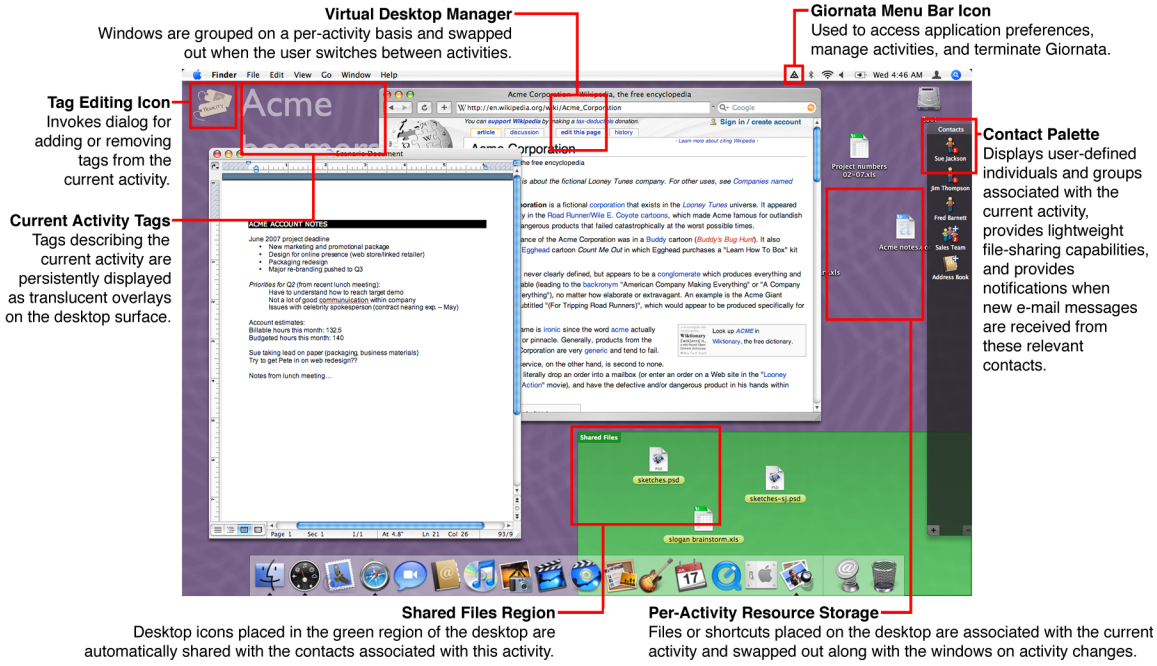


Figure 5.1 The Giornata user interface. In this screenshot, the user is engaged in an activity of capturing notes following a business meeting. There are several tags (including "Acme" and "boomerang"), two open windows, six files (three of them shared), three colleagues, and one group currently associated with this activity.

To provide an overview of the design rationale and implementation of the Giornata system, I first discuss specific requirements for the design of Giornata based on the challenges enumerated in the previous section, provide a scenario that depicts a holistic illustration of the system's support for knowledge work, and conclude with specific details about the interaction design and architecture of the prototype implementation.

5.1 Giornata Design Requirements

The Giornata project explores one facet of activity-based computing: augmentation of the existing desktop metaphor to incorporate representations of activity as a focal aspect of the day-to-day computing experience of a typical knowledge worker. This domain was selected for a number of reasons. First, working in a domain used as the basis for other activity-based computing research efforts provides opportunities to compare and contrast among different approaches that are based on different models of and assumptions about activity. Second, an activity-based system that is designed for the desktop can more easily be built using existing tools and frameworks, which speeds the prototyping process. Finally, a desktop-based system can build upon established interaction metaphors, facilitating the deployment of the system to users and making it easier to gather feedback about its usability and usefulness.

The challenges for the design of activity-based systems described in the previous chapter constitute general guidelines for developing a broad class of interactive systems, both on and off of the desktop. One of the ways to further constrain the design space for the Giornata application is to translate these challenges into more domain- and computing environment-specific requirements, which can then drive detailed aspects of the system design. This process also helps to further constrain the research questions that can be answered using the prototype as an exemplar of activity-based computing systems. In this case, the design requirements will help to focus the subsequent research on understanding

individually-oriented aspects of activity-based computing, such as multitasking, personal information organization, and the integration of communication and collaboration tools as first-class objects in personal activity representations.

5.1.1 Requirements for Supporting Fluid Work Practice

Challenge 1 emphasizes the need for activity-based tools to support multitasking behaviors and to avoid creating additional work for the system's users. Because Giornata builds upon the virtual desktop metaphor popularized by the Rooms system (Henderson & Card, 1986), the system's design will inherently respond to challenges 1.1 and 1.2, since when these challenges are applied to the domain of desktop computing, they correspond closely to the interaction shortcomings that the Rooms system was originally designed to address. As a result, Giornata's requirements for supporting fluid work practice can focus instead on more tightly defining the role of the system as an integrated component of the underlying desktop infrastructure and helping to ensure that the interfaces used to control the virtual desktop aspects of the system necessitate as little interaction overhead as possible during typical use of the system.

> *Requirement 1.* To integrate into existing work practice, the system will provide a unified activity model across all of the applications that users employ, rather than being embedded into a single application.
>
> *Requirement 2.* The system will provide lightweight mechanisms to create, change, and alter activities, since heavyweight interaction techniques and those requiring *a priori* knowledge of how activities will unfold are likely to deter adoption and use.

5.1.2 Requirements for Supporting Multifaceted and Evolving Activities

Challenges related to supporting multifaceted and evolving activities have been only minimally addressed, if at all, in previous research. Therefore, when translating

these challenges into requirements, I have primarily sought to focus them so that they might better serve as design constraints within the context of the desktop-based knowledge work environment.

> *Requirement 3.* Giornata will provide users with tools for informally and formally organizing disparate information within activities. Informal information organization tools will emphasize quick storage and retrieval, without forcing users to explicitly name or find a permanent place for artifacts; formal mechanisms will correspond to long-term storage and retrieval practices.
>
> *Requirement 4.* Real-world activities "overlap" in the way they use artifacts; a given artifact may be used in multiple contexts. Giornata's representations of activity will support this overlap, rather than prescribing that activities be orthogonal or that their artifacts exist in only one context.
>
> *Requirement 5.* Giornata will allow *post hoc* definition of activities, enabling users to map their evolving understanding of the activities into the system; users should be able to create initially unnamed activities and then refine them after the fact. Artifacts used in unnamed activities may need to acquire these refined declarations of use as the activity evolves.

5.1.3 Requirements for Supporting Collaboration Through Activities

The focus of Challenge 3 is in acknowledging the importance of collaboration in modeling activity and the need to design user interfaces that constructively support some of the difficult tasks in coordinating among multiple shared activities without unintentionally disclosing personal or private information along the way. While it would be both interesting and ultimately useful to address this challenge in full in the design of Giornata, so little is currently known about the ways in which users will adopt and utilize activity-based systems that attempting to provide solutions for sharing activities directly

and resolving the differences among individuals' activity representations (per Challenge 3.3) would be a nearly impossible task. Instead, the Giornata system will focus on understanding the ways that individual activity management is taken up when tools are available to support this kind of interaction. This does not minimize the need for activities created in Giornata to support collaboration; however, the design requirements for the system will re-cast some of the collaborative challenges in the context of supporting an individual's collaborative practices within the structure of activity representations. It is my hope that the findings from this dissertation will provide a sufficiently detailed understanding of individual activity-based computing practices to enable future research to engage with some of the more difficult issues in coordinating shared activities across multiple contributors.

Requirement 6. Activities in Giornata will be used as structuring mechanisms for collaboration (i.e., an activity perspective should be integrated into common collaborative tools).

Requirement 7. Because information sharing is a "common case" in accomplishing knowledge work, lightweight sharing capabilities will be integrated directly into the desktop as a first-class interaction technique.

5.2 Interaction Design

Giornata takes as its starting point the virtual desktop metaphor of the Rooms and Kimura systems (Henderson & Card, 1986; MacIntyre et al., 2001; Voida, Mynatt, MacIntyre & Corso, 2002). In addition to providing straightforward activity "spaces" into which focused work on single activities can be concentrated and their constituent components organized, Giornata provides a number of novel information organization and collaboration features.

5.2.1 Scenario of Giornata Use

Bob returns from a business lunch with Acme Inc. and logs into his desktop computer. He immediately switches to the activity tagged with "Acme," "Coyote" and "Boomerang," which automatically populates his desktop with the files associated with the activity, restores the visibility and positioning of relevant open windows, and shuffles the contents of his Contact Palette to display the colleagues and workgroups with whom he has been collaborating to prepare for the meeting. He flips to the word processor document that he has been using to keep notes about his lunch appointment's business account and jots down a few of the outcomes of the meeting.

While Bob is working on capturing his notes, the e-mail icon in his Dock updates to show him that he has received two new unread e-mail messages. Bob resists the temptation to switch over to his e-mail client, knowing that more likely than not, the new e-mails are unrelated to his current task and will distract him from finishing his notes. However, a moment later, the Contact Palette updates to show that one of the messages is from Sue, a colleague associated with the Acme project. He clicks Sue's icon and is taken to a filtered version of his e-mail inbox, displaying only messages that Sue has recently sent. He reads Sue's e-mail and discovers that his boss has already begun asking when a meeting can be organized to review Bob's progress on the Acme account. He quickly finishes working on his notes, saves the file back to the desktop, and then drags it into the shared region of his desktop so that Sue and his boss (both colleagues associated with the activity) can access that file through their corresponding activity workspaces.

Having completed the most pressing business following the meeting, Bob toggles open the overview of his ongoing tasks to take stock of what needs to be accomplished during the rest of the afternoon. Seeing an activity tagged "Home" and "Renovations," Bob remembers that today is the day that he had agreed to draft and send a recommendation letter for a contractor that had recently completed a home improvement project on his house. Rather than closing the windows associated with the lunch meeting,

he simply switches to the other activity. He begins to work on the letter when Jim, another colleague, knocks on his door to determine when Bob might be available to go over an upcoming sales presentation. Bob uses a keyboard shortcut to quickly toggle over to the presentation activity, decides on a meeting time with Jim, and then returns to work. Jim casually asks about the letter and suggests that Bob post his experiences to a local social networking website, "bigvalleyweb." Bob adds the tag "bigvalley" to the activity (automatically tagging the file containing the letter), and logs in to find out where he can post his recommendation online.

5.2.2 Activity-Based Multitasking

In Giornata, each activity is associated with a corresponding virtual desktop. In order to support fluid—and potentially fast-paced—work practices, the system enables users to create a new, empty, unnamed activity using a single keystroke (*per requirements 1 and 2*). This action hides all on-screen windows and desktop contents immediately, presenting the user with a clean canvas in which they can begin work on a new activity without distraction or need to manually manage their digital clutter.

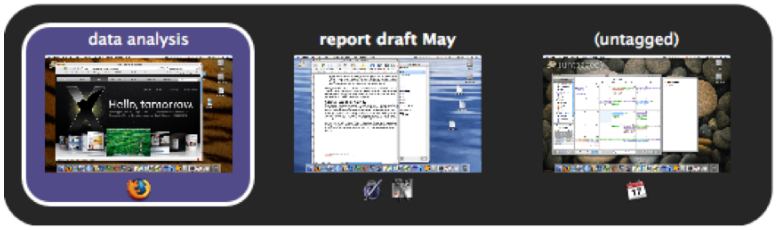


Figure 5.2 The Quick Activity Switcher user interface. The text across the top indicates each of the activities' tags, the thumbnail image is a precise representation of the last state of the activity, and the icons below each thumbnail represent the applications actively associated with each activity.

Giornata allows the user to navigate among open activities using a status bar menu, accelerator keys, or a quick activity switcher (Figure 5.2), depending on their

interaction preference. The quick activity switcher operates using the same user interface principle as the application switching service available both in Windows (invoked using alt + tab) and the OS X operating systems (via command + tab).

Giornata distinguishes itself from previous virtual desktop management systems in three important ways. First, the number of virtual desktops available in Giornata is not fixed as it is in many virtual desktop implementations; users have exactly the number of virtual desktops at their disposal as they have ongoing activities. This prevents unnecessary overloading of virtual desktops and is intended to speed transitions among activities (*requirements 1 and 2*). Second, the objects stored on the desktop surface and the contacts in the Contact Palette transition in and out along with the associated windows. This serves to provide a dedicated storage space associated with the activity and helps to ensure that activities are perceived as cohesive units, including tools, artifacts, and contacts (*requirement 3).* Finally, Giornata allows (but does not require) users to tag activities for quick identification (*requirement 5*).

5.2.3 Activity-Based Resource Storage

In Giornata, the enhanced desktop serves not only as a display space for application windows, but also as an active, ready-at-hand folder for documents and shortcuts associated with the current activity. Any file saved to the desktop (by dragging and dropping or by invoking the standard "save" menu item within applications) is automatically associated with the current activity; as the user switches among ongoing activities, these resources are "swapped out" along with application windows and temporarily stored in dedicated folders associated with each activity until the activity is resumed. The effect of this feature is that the user's desktop workspace is automatically repopulated with the files, folders, and other information resources associated with the current activity, as the user changes activities (*requirement 3)*. This behavior is similar to the approaches taken by Lifestreams (Freeman & Fertig, 1995; Freeman & Gelernter,

2007) and Time-Machine Computing (Rekimoto, 1999), with the main difference being the underlying organizing principle that determines the visibility of the desktop's contents—ours being activity and the others', time.

These capabilities scope the information displayed on the screen at any one time to the most relevant applications, information resources, contacts, and communications when the user is immersed in a particular activity (*requirements 1 and 3*).

5.2.4 Activity Tagging

Each activity created in Giornata can be annotated with optional, freeform tags to describe the semantics of the activity. Activities are initially created without tags; the ability to create and work in an unnamed desktop allows work to proceed even in cases in which the user might not know the significance or eventual meaning of the activity at its outset.

An activity's tags are displayed throughout the Giornata user interface to help users identify the activity in which they are currently working and to distinguish among background activities. The active activity's tags are persistently visible, rendered in a large, translucent font over the desktop wallpaper. For users with limited desktop screen real estate (e.g., laptop users) or those who frequently work with full-screen windows, the tags can also be displayed in the status bar (the right portion of the menu bar stretching across the top of the screen) next to the menu item that Giornata installs for switching among activities, setting application preferences, or shutting the system down.

When an activity has one or more tags associated with it, these tags are transferred to each file that the user touches over the course of working in that activity. This design serves to "stamp" files with information about the context in which they were created or edited and helps to overcome the burdensome process of manually adding semantic metadata to each individual file associated with an activity, a similar approach to that taken by Dourish et al. (2000). It also allows documents that are shared across

multiple activities to "inherit" the tags of all the activities. Because the Spotlight framework automatically indexes these tags, users can quickly find information resources using the semantically meaningful tags that they, themselves, assigned to the activity, regardless of where the files associated with the activity are actually stored on the disk (*requirements 3 and 4*).

As the user comes to understand the meaning of a particular activity, she can edit the activity's tags by clicking on a tag icon on the desktop surface. She is then given the option to simply tag the activity from that point forward, or to retroactively tag all of the files previously associated with the activity as well (Figure 5.3). This ability to create post hoc tags on activities and files enables users to refine the meaning of an activity as that meaning emerges over the course of accomplishing the work. It also helps to ensure that the system's activity representations are sufficiently flexible to adapt to the user's evolving work environment (*requirements 2 and 5*).

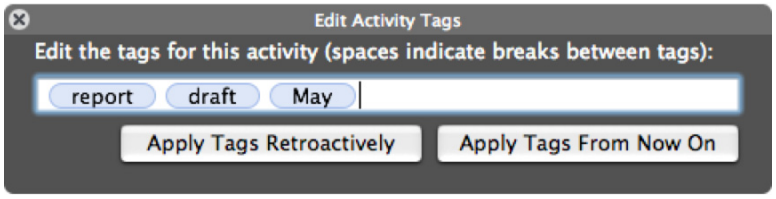


Figure 5.3 Giornata's tag editing user interface, revealed by clicking the tag icon persistently displayed on the user's desktop. Changes to an activity's tags can be applied at a particular point in time or retroactively; the latter option attempts to re-tag all of the files previously tagged over the activity's lifetime using the new set of tags.

5.2.5 Activity-Aware Collaboration Support

Giornata provides two features to support activity-aware collaboration. First, Giornata integrates a subset of the sharing palette interface (Voida, Edwards, Newman, Grinter & Ducheneaut, 2006) to enable lightweight collaboration. This "Contact Palette" component, attached to one side of the display space, provides a persistent visual

summary of those colleagues and groups the user has associated with the current activity (Figure 5.4). Giornata allows the user to drag representations of colleagues and groups from the system's Address Book application into the Contact Palette, an action that "binds" the contact to the current activity. Like the open windows and files stored on the desktop, contacts are "swapped out" as the user transitions between activities, providing a persistent display of the contacts most salient to the current task (*requirement 6*).

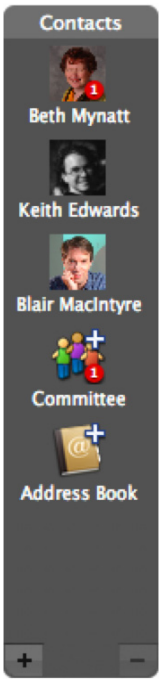


Figure 5.4 The Contact Palette component of the Giornata user interface. The icons represent three colleagues and one ad-hoc group associated with the current activity; the numbered, red "badges" indicate that the user's e-mail inbox currently contains one unread e-mail from the contact "Beth Mynatt" and one from a member of the "Committee" group. The "Address Book" icon reveals contacts from the user's OS X Address Book, allowing drag-and-drop association of contacts with the activity.

The Contact Palette provides a number of awareness and collaboration services. A popular convention among e-mail clients developed for the OS X platform is to display a "badge" on the e-mail application's Dock icon to indicate the presence and number of

unread messages in the user's inbox. Although this provides a convenient, at-a-glance summary of pending communications, it provides no context about the source or relative importance of these incoming messages; new e-mail could just as easily be irrelevant or distracting ("spam") as important or useful. The Contact Palette periodically connects to the user's e-mail client and annotates individuals and groups in the palette with "badges" indicating the number of unread e-mails originating *from that person or group*, providing a summary of unread e-mails that are more potentially relevant to the current work context.

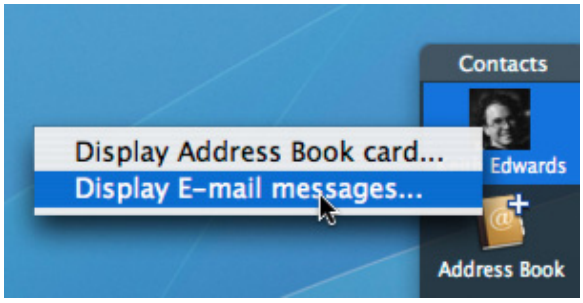


Figure 5.5 The Contact Palette features multiple options for quickly accessing information about the colleagues associated with the current activity; these options can be arbitrarily expanded to provide relevant resources for a given organizational context.

Because the icons representing individual colleagues are connected to their corresponding Address Book cards and e-mail addresses, Giornata users can click on an icon to quickly access a variety of information about the contact. In the current system's implementation, options include displaying the Address Book card for the contact or a filtered view of the e-mail client's inbox, showing only messages sent by the selected contact (Figure 5.5). However, these options could be expanded to reveal any type of information that might be useful in a given organizational context. The Contact Palette can also be used to share files with the individuals who are associated with particular activity using the same interaction design used in the sharing palette user interface (Voida

et al., 2006). Files can be dragged and dropped directly on the Contact Palette to share a file with a particular contact or group (*requirement 7).*

Giornata's desktop also includes a "shared files" region, which provides a persistent, spatial connection among collaborators' activity desktops. When files are dragged into this region, they are automatically replicated on each of the collaborators' desktops and updated each time the files' contents are changed. This region acts as an information-sharing portal across all collaborators' desktops, but also allows all participants in an activity-based collaboration to control, as is contextually appropriate, the degree to which information is shared (*requirements 6 and 7*).

Several prior systems have attempted to use activity as a means for fostering collaboration, including UMEA (Kaptelinin, 2003) and ActivityExplorer (Geyer, Vogel, Cheng & Muller, 2003; Muller, Geyer, Brownholtz, Wilcox & Millen, 2004). However, the main distinction between these systems and Giornata is that Giornata integrates both the activity representations and the collaboration tools into the desktop interface itself; the others rely on interaction within a standalone application.

5.2.6 Supporting Implicit and Explicit Interactions

Giornata's user interface integrates closely—and nondestructively—with the existing file and window management components of Apple OS X (*requirement 1).* The stock OS X window manager emulates the physical manipulation of paper on a desk by compositing application windows on various layers above the desktop file icons and wallpaper, but below system-wide interaction widgets like the menu bar and the Dock (Figure 5.6). Giornata augments this visual stack by inserting two additional layers: an explicit interaction layer on top of all other layers (Figure 5.6a), providing persistent visibility of the Contact Palette and allowing users to control the activity management system, and an implicit interaction layer below the desktop file icons but above the background wallpaper (Figure 5.6e). This non-interactive layer serves as a persistent

information display for low-priority information such as the current activity tags. It also passively monitors the user's interactions with existing desktop objects (such as desktop file icons), providing the system with input as a side effect of other, typical desktop interactions.

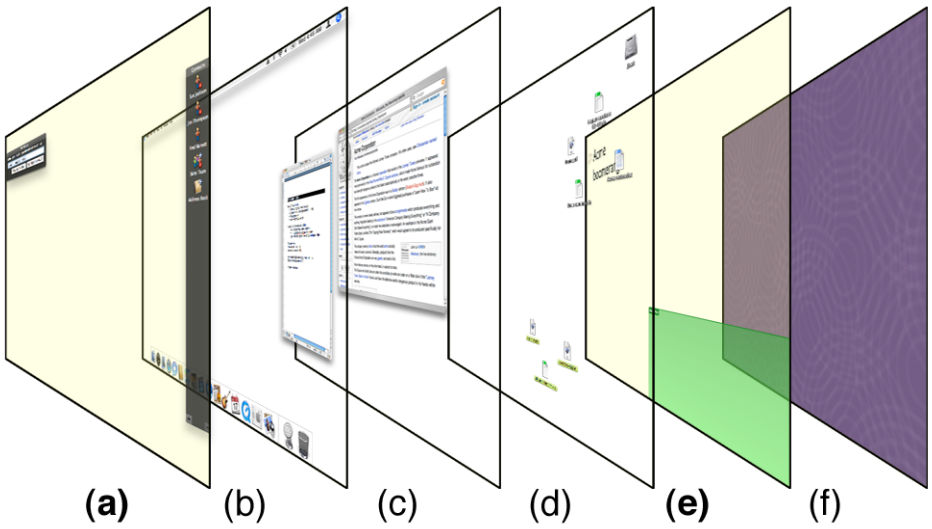


Figure 5.6 Explicit and implicit interaction layers in the Giornata system and their relationship to existing window manager interaction layers. This figure illustrates the interaction layers of Figure 1: (a) Giornata's explicit interaction layer, including activity management dialogs and the Contact Palette; (b) the system menu and Dock; (c) application windows; (d) desktop icons; (e) Giornata's implicit interaction layer, including activity tag display and sharing space; and (f) the desktop wallpaper.

The implicit interaction layer is a particularly powerful component of the Giornata user interface design. Because it serves as a persistent information display and is "anchored" to the desktop wallpaper and rendered translucently, a quick overview of the activity state can quickly be surmised by invoking the "show desktop" feature of Exposé. The seamless augmentation of the desktop background also helps to convey Giornata's status as an integral part of the desktop environment.

Additionally, the implicit interaction layer, together with a filesystem change-monitoring daemon, serves to manage the public/private sharing status of desktop items based solely on their location on the desktop. A Giornata user can indicate that a file associated with an activity is to be shared freely with relevant colleagues simply by adding those colleagues to the Contact Palette and moving the file to a "shared files" region rendered by the implicit interaction layer. Giornata automatically notifies the colleagues whenever files are added to this region or existing files in that region are changed. Likewise, dragging the file icon out of the shared file region and dropping it elsewhere on the desktop suspends further automatic sharing of the file.

5.3 System Architecture

Giornata has been implemented on the Apple OS X operating system as a hybrid Carbon/Cocoa/AppleScript-based application. The application is designed to run continually while a user is logged in and provide activity-management services alongside, and essentially independent from, other system applications.

I chose OS X as the host platform for the Giornata prototype for three main reasons. First, the OS X window manager already provides a framework (albeit undocumented) for creating and managing virtual desktops. Second, Apple's use of a metadata-enabled filesystem (HFS+), tightly integrated with its Spotlight search engine, enabled me to create a robust file- and activity-tagging infrastructure that could integrate easily into users' existing information foraging practices. Third, AppleScript, a powerful cross-application scripting language integrated into the OS, allowed me to quickly prototype interactions with existing applications and data sources without need for modifying other applications' source code to be explicitly "Giornata-aware."

Although Giornata is technically just another application running on the system, it is designed to integrate as closely as possible into fabric of the underlying operating system. This degree of integration is accomplished in part by building upon high-level

OS services, which ensures that activity-related actions within are immediately reflected in other applications and the operating system, itself (Figure 5.7). This design creates an activity-centric user experience while allowing the user to run existing applications alongside Giornata without penalty or modification.

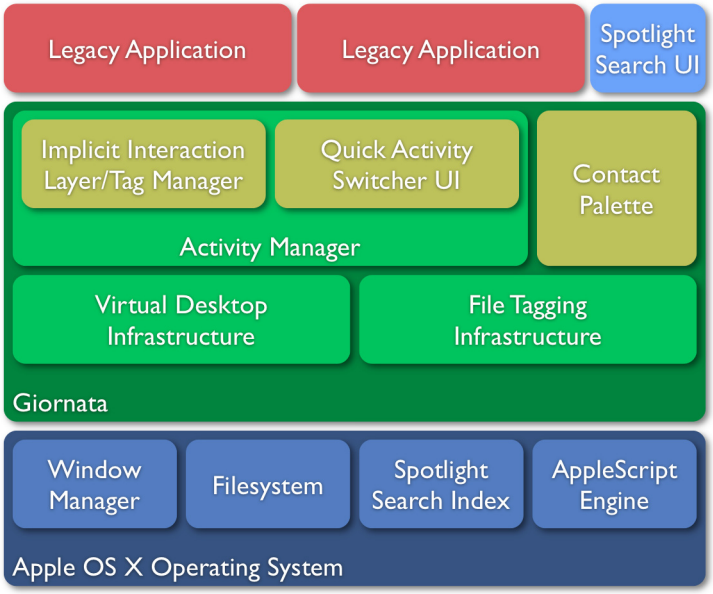


Figure 5.7 A high-level overview of Giornata's system architecture. The prototype builds on several core system services and presents itself as an integrated component of the operating system, enabling representations of activity within existing applications without requiring that these applications be modified.

In the following sections, I unpack the implementation details of each of Giornata's main program modules, focusing on the ways in which each take advantage of the Cocoa programming frameworks and draw on lower-level system services and APIs to accomplish the goals of the system's interaction design.

5.3.1 Virtual Desktop Infrastructure

In order to implement the core virtual desktop functionality, Giornata builds on an existing-but-undocumented API supporting multiple virtual workspaces that has been

present in OS X for the last several years, but only recently (officially) utilized by Apple's Spaces[1] feature in OS X version 10.5 ("Leopard"). This API provides functions to query for the current virtual workspace; perform an optional, animated transition between workspaces; get and set the virtual workspace on which a particular window appears; and set some extended attributes on windows, such as whether or not they are immune to Exposé's window management tools.

Because these function calls are not exposed in the publicly distributed Carbon/Cocoa header files in the OS X software development kit, implementation of these features requires a supplementary C header file defining the syntax and format of the virtual desktop functions. Additionally, some of the capabilities needed to implement a fully functional virtual desktop system (e.g., the ability to move windows from one workspace to another) are available only when executed in the same process context as the core window server. In order for these calls to succeed, Giornata takes advantage of a feature of the Mach kernel known as *code injection*. Using this technique, the code of a running process—in this case, the application code for the Dock application, which also serves as a host for a particular user's instance of the window server—is dynamically modified to include additional code that listens for inter-process communication requests (via AppleScript's underlying Apple Events messaging mechanism) from the main Giornata application to read or write various virtual desktop-related settings in the window server that would otherwise be unavailable to a user-space application like Giornata.

Fortunately, much of the source code necessary to realize the virtual desktop component of Giornata had already been written by others in the Macintosh software development community and disseminated under various open source licenses. Giornata's virtual desktop code is derived in large part from an open source virtual

[1] http://www.apple.com/macosx/features/spaces.html, accessed 15 January 2008

desktop application named VirtueDesktops[2] (colloquially, "Virtue"), which in turn evolved from a project named DesktopManager[3]. DesktopManager's developer, Rich Wareham, was responsible for reverse-engineering Apple's private virtual workspaces API and generating the supplementary header file. The other significant open source project that Giornata's virtual desktop manager builds upon is a C-language module by Jonathan Rentzsch known as *mach_inject*[4], which provides a straightforward mechanism for injecting the requisite code for managing windows' extended attributes into the Dock application's process context.

5.3.2 File Tagging Infrastructure

Just as Giornata's virtual window manager implementation makes use of included-but-previously-unutilized functionality within OS X's window server, its file tagging module leverages a capability of OS X's default HFS+ filesystem that is also largely unexposed to application developers. As of version 10.4 ("Tiger"), OS X's system-wide search infrastructure, Spotlight, automatically indexes all files on the user's computer based on their filenames, contents, and other file attributes, such as creation time. While files' contents and basic attributes are easily modified using standard file I/O programming techniques, Spotlight also indexes a variety of extended attributes stored on each file, which are (currently) less easily manipulated by applications. One of these extended attributes, "Spotlight Comments" (also referred to as "Finder Comments"), can be used to annotate any file on the computer with a standard, UTF-8 encoded string. Although this comment attribute is easily viewed or edited from a file's "Get Info" dialog in the OS X Finder, there is no documented mechanism enabling third-party applications to make changes to the contents of the field directly. Instead, applications can "ask" the

[2] http://virtuedesktops.info, accessed 15 January 2008
[3] http://desktopmanager.berlios.de, accessed 15 January 2008
[4] http://rentzsch.com/mach_inject/, accessed 15 January 2008

Finder to read or write the contents of the attribute on their behalf using a short AppleScript.

Giornata's tag manager is implemented as a set of extensions to Cocoa's NSFileManager class, providing additional functions for setting and retrieving Spotlight Comments for specified files via AppleScript, for parsing comment strings into tag lists, and for encoding tag lists as comment strings. Although few mainstream applications on the platform currently use the Spotlight Comments attribute to store any data of value—until recent versions of the operating system, it was relatively easy for the contents of this attribute to be lost when files were moved or copied across media—Giornata takes care to preserve any existing contents of the comments string when adding tags to a file and also uses a distinctive encoding scheme so that it can distinguish between its own tag data and other, non-Giornata information that may co-exist alongside the tags. All Giornata tags are prefaced with an "@" character and appended to the end of the non-tag comment string using a space character as a tag delimiter. This encoding scheme is computationally straightforward, ensuring that the system can quickly read or write tags for a large number of files without incurring significant overhead. It also provides a human-readable representation of the tags that can be viewed or edited using the Finder (see Figure 5.8) or used as search keywords in Spotlight. (Spotlight operates based on a substring search algorithm that operates on a per-character, rather than a per-word basis; as a result, files whose Spotlight Comments attribute contains the encoded tag substring "@dissertation" are considered positive matches for Spotlight searches using the keyword "dissertation.")

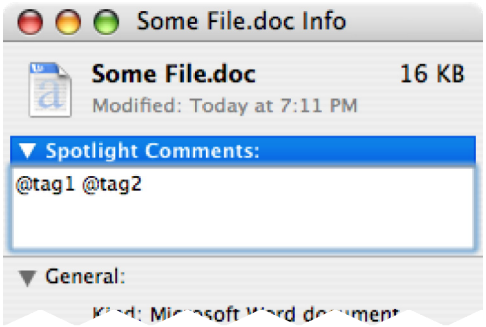


Figure 5.8 Activity tags are automatically applied to files modified over the course of work in an activity. These tags are encoded and stored as human-readable elements in each file's Spotlight Comments extended attributes field and easily accessed in the Finder.

When Giornata starts up, it launches a helper daemon process to monitor and automatically apply tags to each file that the user touches. This process, running with root-level privileges, takes advantage of the *fsevents* kernel-level filesystem monitoring used by Spotlight to detect the creation and modification of files throughout the filesystem. (By default, the daemon only takes action when a file change is detected within the active user's home directory.) This approach ensures that all file interactions are promptly noticed by Giornata and allows the system to automatically tag files as soon as files are touched and without requiring extensive hooks to be added to existing applications.

When the helper daemon notices that the user's desktop database file ("/Users/*username*/Desktop/.DS_Store") has been modified, indicating that items have been added to, removed from, or moved to a different location on the user's desktop, the daemon sends a distributed notification to the main Giornata application indicating that some interaction has taken place that might be considered implicit input for the system (e.g., a file has been moved to or from the sharing space on the desktop). This notification is handled by Giornata's implicit interaction layer, described in detail below. The helper daemon also subscribes to receive distributed notifications of activity switches from the main Giornata application process, ensuring that the tags that it applies to changed files

correctly reflect the activity that is "open," or foregrounded, at any point in time. A short pause in tagging is triggered upon receipt of an activity switch, helping to minimize spurious tagging of files touched just prior to or immediately following an activity transition.

5.3.3 Activity Manager

The Activity Manager serves, in essence, as the heart of the Giornata system. This component, implemented as the main application process, is responsible for launching and populating the virtual desktop infrastructure, starting up the file tagging helper daemon process, hosting the centralized activity model, coordinating among Giornata's various user interface components, and performing mundane administrative tasks, such as maintaining user preferences and automatically checking for updated versions of the software. Much of the Activity Manager code is derived directly from VirtueDesktops' application code, the main differences being a series of enhancements to the activity model necessary to support the unique aspects of Giornata's interaction design (relative to other virtual desktop managers) and a dramatic simplification of the rather complex plug-in–based architecture used in Virtue to encourage third-party development around its virtual desktop platform.

Virtue's stock "activity" model reflected its focus on providing virtual desktop services. The original model was a list of virtual desktops, each corresponding to and serving as an application-specific proxy for a virtual workspace managed internally by the OS X window manager. These virtual desktop structures also stored application-specific data including a user-friendly display name, the path to the desktop's custom wallpaper (if specified), a list of windows currently displayed on that desktop (also proxy objects for "real" windows managed by the OS X window manager), and a list of all applications that own one or more of those windows (in order to enable functionality binding an application to a particular virtual desktop). Virtue also maintained a data

structure (separate from the primary activity model) mapping the virtual desktops to row and column positions in a table, allowing wireframe renderings of all virtual desktops to be displayed in a traditional overview grid (similar to Apple's Spaces overview).

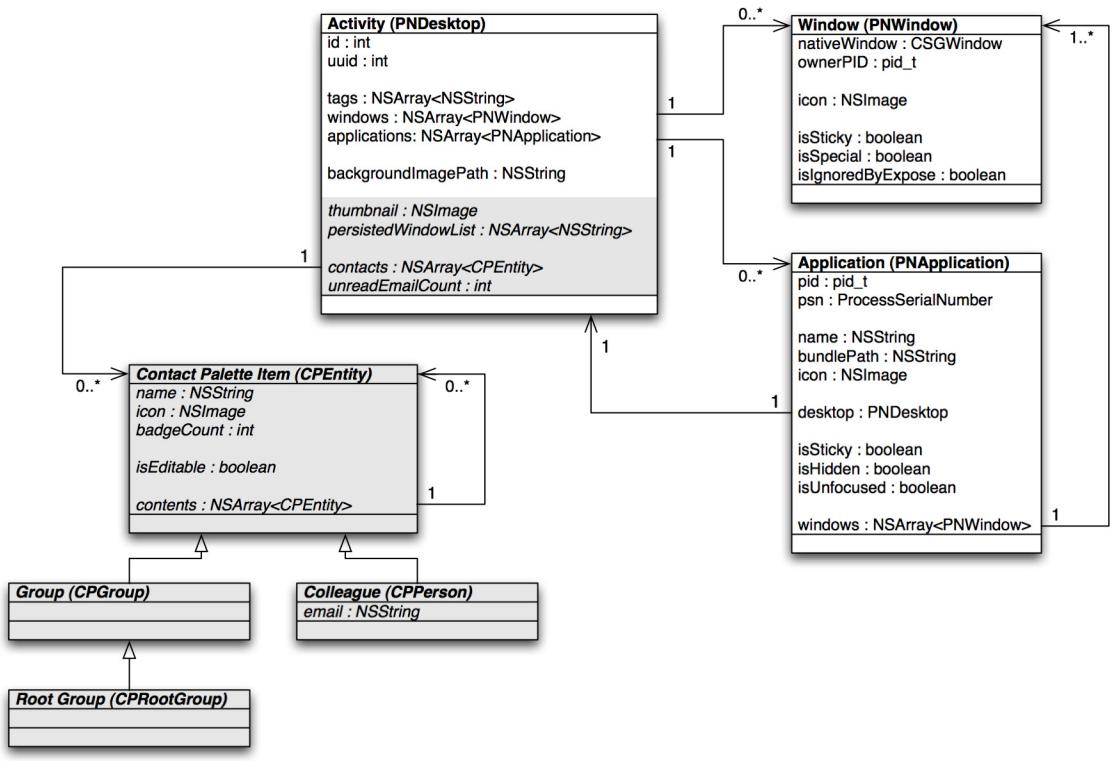


Figure 5.9 Giornata's internal activity model, illustrated as an UML entity-relationship diagram. Shaded areas represent Giornata's additions to VirtueDesktops' more virtual desktop-centric data model.

Giornata's activity model extends Virtue's in five significant ways (see also Figure 5.9). First, Giornata's activities substitute a list of activity tags for Virtue's user-friendly display name; in Giornata, the tags (if present), represent the user-readable identifier for the activity. These tags are displayed in place of the display name in Giornata's user interface elements and can overlap from activity to activity to represent semantic commonality among related tasks. Second, each activity in Giornata's model includes a list of contacts related to each activity, elevating the notion of individuals to

the same first-class status in an activity that windows and applications hold in traditional virtual desktop managers. These contacts are represented in a tree structure and can consist of individuals, identified by their e-mail addresses, or arbitrarily specified groups. This contact data structure also includes storage for awareness information, such as the number of e-mails sent by each individual (or, for groups, the number of e-mails sent by all members of the group) that have been received but not read in the user's e-mail inbox. Third, activity objects have a new method for creating and maintaining a folder in the user's home directory (typically "/Users/*username*/Activities/*activity tags*") where desktop items are moved when the user switches away from or closes the activity. Fourth, each time an activity transition takes place, the system captures an image of the user's desktop before swapping out any of the desktop contents and stores this image in the activity's data structure. This image, analogous to Kimura's "preserving spatial relationships" montage rendering technique, provides a visual representation that can be used both to quickly identify the activity and the state in which it was left the last time it was active. Finally, Giornata does not store any kind of row or column positioning data with activities that could be used to position them in a grid, because the number of activities open at any point in time may vary. Instead, Giornata simply uses the activities' order in the activity list to provide each with a persistent location in a linear display list (e.g., left-to-right in the quick activity switcher or top-to-bottom in the status bar menu, illustrated in Figure 5.10). In future versions of the system, it might be desirable to augment the linear ordering of Giornata's activities with an (X, Y) position coordinate and scaling factor in order to provide a Kimura-like overview of all open activities.

The quick activity switcher user interface is a straightforward, linear visualization of ongoing activities (Figure 5.2). This interface directly extends the "desktop pager" component from the original VirtueDesktops implementation, and consists of a single, semitransparent Cocoa window hosting a variable-sized matrix of custom user interface "cells," each representing a single activity.

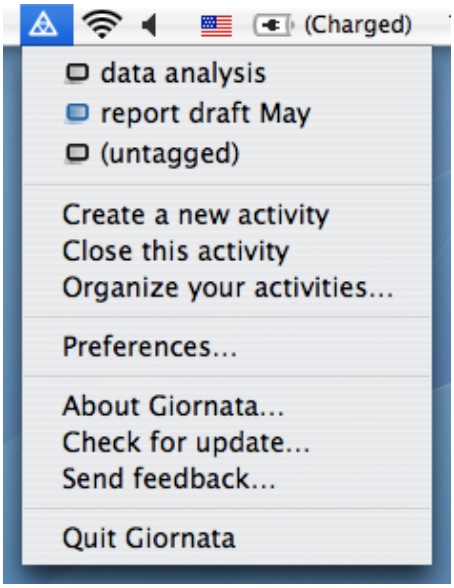


Figure 5.10 The status bar menu maintained by Giornata's activity manager. This menu can be used to quickly jump between open activities or to perform other administrative functions; the order of the activities at the top of the menu can manipulated by users via the quick activity switcher user interface.

5.3.4 Implicit Interaction Layer/Tag Manager

Giornata's implicit interaction layer is implemented as two windows, manually placed near the bottom of the window sorting depth stack, or "z-order," just above the desktop background (see Figure 5.6e). The lower of these two windows is a borderless, semi-transparent, full-screen window that is excluded from receiving input focus events (e.g., mouse clicks and key presses). This window serves primarily as a persistent display for Giornata, providing a canvas on which the current activity's tags are rendered in a large font and delineating the green region of the screen in which documents are shared with an activity's collaborators. The second window, displayed "above" the first in the z-order, is a smaller, borderless, interactive window that appears as a tag icon. When the user clicks on the tag icon, a tag editor dialog is displayed as a normal, "HUD-style" utility window—inspired by the palettes used in Apple's iPhoto and iMovie applications—at the usual location at the front of the z-order (see Figure 5.3 for an

example of the tag editing user interface and Figure 5.6a for its position in the window ordering stack.) The two implicit interaction layer windows are both annotated with a window style bit that prevents the Exposé window management software from manipulating the windows, ensuring that they remain fixed in position, even when the Exposé functions to display all open windows as thumbnails or push all open windows aside to reveal the desktop are invoked.

The implicit interaction layer operates as a semi-autonomous component of the Giornata applications. The Objective-C classes associated with handling rendering tasks and interaction events for this layer also subscribe to receive distributed notifications of three types: activity switch notifications, tag change notifications, and desktop item content changes. When the activity manager announces an activity switch or tag change, the implicit interaction layer retrieves the new set of tags and refreshes its desktop tag display. When the file tagging infrastructure posts a notification about the desktop contents changing (see section 5.3.2 for details), either as a result of an explicit user action or as a side effect of an activity switch, the implicit interaction layer examines each of the items on the desktop using a small AppleScript to determine if its physical positioning falls within the boundaries of the sharing space. If it does, the AppleScript automatically toggles the item's Finder highlighting on (as a confirmation that the system has recognized and begun sharing the item) and, once the sharing status of all desktop items is determined, a message is passed along to the Contact Palette indicating the current list of files to be made available to all colleagues associated with the activity.

5.3.5 Contact Palette

Like the quick activity switcher, the Contact Palette is implemented using the standard model-view-controller paradigm atop Apple's Cocoa development framework. The palette operates as a semi-autonomous component of the Giornata system, providing a visual representation of the individuals and groups associated with the current activity

using a series of semi-transparent, HUD-style windows and arrays of custom, icon-centric widgets to mimic the user interface of the right-hand panel of the sharing palette (Voida et al., 2006). Each of the palette windows are also managed by an animation framework that smoothly displays and hides sub-palette windows when needed (similar to the canonical pop-up menu interaction) and can optionally slide the main palette window mostly off-screen when it is not being used, in order to minimize Giornata's screen real-estate footprint while keeping information about relevant colleagues close at hand.

The Contact Palette maintains awareness about the user's current activity by registering for distributed notifications from the activity manager. When an activity transition is detected, the palette synchronizes its contents with the outgoing activity's representation in the central activity model and then retrieves the contact information for the incoming activity and updates its display. When the Contact Palette receives a notification from the implicit interaction layer that a file has been moved into or out of the sharing space on the desktop (per section 5.3.4), the palette passes the full path of the file and the e-mail addresses of the contacts currently associated with the activity to a peer-to-peer file sharing library originally developed for the sharing palette prototype, which manages the process of replicating the files across the network.

Additionally, the Contact Palette serves as one of the key bridges between Giornata and the surrounding desktop "ecosystem," connecting to a number of external services and applications. The palette uses the OS X Address Book framework to dynamically update the list of contacts that can be associated with an activity. Any contact in the user's Address Book database with an e-mail address is automatically added to the "Address Book" group, available at all times at the bottom of the Contact Palette user interface. The palette also connects to the user's e-mail client periodically via AppleScript, retrieving a list of all unread messages in the inbox and updating the Contact Palette individual and group icons with badges that indicate the availability of

new communications from colleagues deemed relevant to the current activity. Finally, when a user clicks on an individual colleague's icon in the Contact Palette and selects one of the collaboration-focused options from the pop-up menu (Figure 5.5), the palette executes an AppleScript corresponding to the selected menu item—either to display the contact's Address Book card in full or to activate the user's e-mail client window and display a filtered view of the inbox, showing only those messages sent by the contact. Because most current OS X applications are AppleScript-able, the degree of integration demonstrated by the Contact Palette in the Giornata prototype represents just a few of the possibilities in integrating the software into the larger desktop and application environment.

5.4 Implementation and Deployment

The Giornata system was developed over the course of several months and resulted in several incremental release milestones for testing, public demonstration, and, eventually, deployment purposes. Over the course of the development process, the Contact Palette component was the focus of some of the most significant revisions, both in order to improve the consistency and reliability of its largely custom user interface implementation and because of its extensive connections to other applications and operating system components. Other major revisions to the software that were enacted based on feedback elicited from the pilot and deployment releases included:

- adjusting the system's file tagging behavior to tag files only within the user's home folder (although not the home folder itself) and to avoid tagging invisible files and the internal contents of OS X file "bundles"—directories that are treated like single files by the Finder;
- improving crash recovery in a variety of ways, including writing the list of window–activity associations to disk frequently to make the process of repopulating activities on a recovery restart less onerous and using a

temporary file as a flag to indicate whether the last execution exited cleanly to help the system determine whether or not items stored on the desktop at startup should be implicitly associated with a known activity or moved out of the way;

- adding superficial support for multiple-monitor configurations and some code to handle display configuration changes gracefully;
- expanding the number of e-mail clients with which the Contact Palette is able to connect;
- implementing a "presentation mode" that prevents the Contact Palette from revealing itself and forces the implicit interaction layer to render as a completely transparent layer while the computer is connected to a projector;
- streamlining some of the inter-application communication code to run in separate threads and prevent blocking on the user interface (effectively freezing the entire system); and
- improving the logging output of the system for data collection purposes.

At the end of the deployment phase, Giornata's code base consisted of 89 Objective-C class files (and their associated headers), 14 C-language source files (and their associated headers), and 13 user interface description files. (Some of the object-oriented programming relationship information is "freeze-dried" into the user interface description files when programming within Apple's Objective-C/Cocoa paradigm.) Additionally, based on heavy developer testing and participant feedback, 69 unique implementation bugs and feature requests had been logged at the conclusion of the study for consideration when developing future versions of the system.

CHAPTER 6

EVALUATION OF THE GIORNATA SYSTEM

The design of Giornata reflects several years of background research in the domain of activity-based computing. Many of the program's features were directly influenced by observations about the ways that knowledge workers interact with computers and theories about the ways that humans cognitively process and manage activities. Although the realization of a system like Giornata represents a significant research contribution in its own right, I have also undertaken a user study based on an extended deployment of the prototype system. This study served two primary purposes: (1) to elicit feedback about the user experience provided by this particular instantiation of an activity-based computing environment, and (2) to gain an understanding of knowledge workers' initial impressions about working in a computing environment where activity is the primary structuring principle. Because the study was designed to be dual-purpose, the findings serve both as an evaluation of Giornata, as a discrete software artifact, and activity-based computing, as an interaction paradigm.

I deployed a prototype version of the Giornata software for *in situ* use. Five participants used the software for an average of nearly two months. In this section, I present quantitative results gleaned from surveys and log data collected by the Giornata software as well as qualitative results from interviews carried out over the course of the deployment.

6.1 Research Questions and Hypotheses

The user study was intended to elicit feedback on the usability and usefulness of the Giornata system. However, I also wanted to use the study to capture broader impressions about a desktop computing environment that features activity as the primary organizing principle. The following four research questions were used to define the scope

of this user study and to guide the design of the data collection instruments that were used. After each research question, I describe my initial hypotheses about how the participants would respond, based on previous empirical studies of knowledge work, cognitive theories of activity, and the particular design decisions embodied in the Giornata prototype.

Research Question 1. *Are discrete representations of activity sufficient for representing and supporting the complexity of knowledge work?*

Hypothesis 1.1. Participants will create a variety of activity representations over the course of the deployment and the majority of these activities will be transient in nature, representing work completed in a short, finite amount of time. This hypothesis is grounded in the objective nature of activity (Leont'ev, 1978; Vygotsky & Cole, 1978, Engeström, 1987), past empirical studies of knowledge work emphasizing its fracturing nature (e.g., Sproull, 1984; González & Mark, 2004), and Giornata's intentional design emphasis on reducing barriers to managing activity (section 5.1.1).

Hypothesis 1.2. Documents will be associated with individual activities in all but a few isolated cases. This hypothesis is based on the centrality of information processing to knowledge work (Kidd, 1994) and studies identifying activity as a key index for organizing documents in the workplace (Bergman, Beyth-Marom & Nachmais, 2006).

Hypothesis 1.3. "Metawork" tools (e.g., e-mail, IM) will be shared across multiple activities, with a higher precedence of sharing taking place within higher-level tasks and much lower precedence of sharing taking place within shorter-term, transient tasks. This hypothesis is based on previous empirical work that identified the important role of "metawork" in activity (González & Mark, 2004) and the results of my previous study (section 2.2; Nair, Voida & Mynatt,

2005), which reiterated the complexity of incorporating general-purpose communications tools into activity-oriented work clusters.

Research Question 2. *How does the availability of informal tagging affect resource organization strategies on desktop computers?*

Hypothesis 2. Participants will spend less time manually filing documents, relying more on the per-activity storage facilities provided by Giornata. My intuition is that a gradual transition to this style of resource organization will be seen over the course of the deployment. This hypothesis is based on previous empirical work by Bergman et al. (2006) identifying the importance of activity for classifying information, but also from Card, Pavel & Farrell's original model of windows working sets (1984): if virtual workspaces can reduce the time spent organizing windows, the same premise should hold true for the documents that are associated with those "working sets."

Research Question 3. *Is informal tagging sufficient for recording a user's evolving understanding of an activity? Are these tags a useful index for finding relevant resources?*

Hypothesis 3.1. Participants will apply a small but descriptive number of tags to each activity, adding them incrementally over the life of the activity. This hypothesis draws from Kidd's study of knowledge workers, which emphasizes the importance of the meaning-making process (1994). My intuition is that participants will use the tagging feature conservatively both because they are primarily tagging the activities for their own benefit, requiring less semantic labeling to be personally meaningful, and because the information organization tools with which these participants will be most familiar (e.g., the hierarchical filesystem) currently support relatively terse labeling facilities.

Hypothesis 3.2. The degree to which tags are used will correlate with the participants' self-reported use of Spotlight search in finding documents—the

more the participant reports using Spotlight, the more they will utilize the tagging feature to organize their files. This hypothesis is primarily grounded in common sense intuition about the role that Spotlight plays in information organization and the design focus of Giornata in supporting a combination of activity- and search-based information retrieval (section 5.1.2).

Research Question 4. *Do representations of activity serve as a useful constraint/boundary for managing collaborations?*

Hypothesis 4. Participants will readily identify the individuals who are relevant to each activity and will find it useful to filter communications and share files using this list of relevant colleagues. This hypothesis is grounded both in theories of activity that underscore the importance of the surrounding community in accomplishing an objective (Engeström, 1987) and in empirical studies about the influence that colleagues have on work (e.g., González & Mark, 2004) and information sharing practices (Olson, Grudin & Horvitz, 2005).

6.2 Research Design

In order to answer these research questions, I considered several possible approaches for evaluating the Giornata system. Some of these initial research proposals focused on running a series of controlled laboratory studies, designed to gather data about use of activity-based systems in the context of constrained, concrete scenarios. This style of research design would have greatly simplified the implementation of the prototype system—only those components needed to carry out the study would have needed to be mocked up in full—and would have enabled data collection and analysis contrasting multiple participants' responses and observed interaction strategies based on a common situation. However, this lab-based approach was rejected in favor of deploying a fully-implemented system to be used in the course of everyday work for two reasons: a need to

evaluate the system in an ecologically valid context and a need to evaluate the system over an extended window of time.

Transforming information is central to knowledge work (Kidd, 1994); the process of categorizing information into semantically meaningful clusters (activities) constitutes a significant portion of the intellectual labor involved in making sense of information and transforming it. Because this kind of intensive information handling requires both an investment in and deep understanding of the background of the information being processed, a study of activity-based practices in an artificially constructed scenario with information that the participant has not seen before would seem to provide superficial results, at best, and spurious results, at worst.

Additionally, an ecologically valid study would also need to preserve another key aspect of activity: the tools with which the activity is accomplished. An acknowledged characteristic of knowledge workers is their diversity, both in terms of their output and the ways in which they manipulate information (Kidd, 1994). Modern knowledge work might involve any number of applications, online tools, databases, or personal document or communication histories; replicating this kind of diverse foundation from which activity-based behaviors might emerge in the context of a laboratory study would be virtually impossible.

Making meaning out of information is a process that requires—and takes place evolutionarily over—an extended period of time. Understanding activity-based systems' role in knowledge work requires an examination of how participants adopt and appropriate the system over time. A time-constrained lab study would also be an inappropriate evaluation for examining the effects of activity-based information organization on longer-term information retrieval strategies or the iterative development of working practices around the capabilities (or limitations) of the system instantiating activity-based concepts.

6.3 Deployed System Details

The version of the Giornata software at the center of the user study was a fully functional activity-based prototype. Because the primary goal of the study was to understand how people would fundamentally use an activity-based system in the course of everyday knowledge work, I opted to focus on ensuring that the core activity-centric features of the system were as stable as possible for the long-term, *in situ* evaluation.

Gaining insights about the collaborative uses of activity-based systems was a secondary goal of this research. For the study, I selected a subset of the system's collaboration features to deploy to participants, in the interest of balancing between providing a reasonably stable system to the participants, on the one hand, and being able to explore the ways that a breadth of potential collaboration features would be used, on the other.

Based on my previous experiences in developing activity-based systems (MacIntyre et al., 2001; Voida, Edwards, Newman, Grinter & Ducheneaut, 2006), I deemed that document sharing and e-mail awareness would likely be the most critical collaboration tools for the system to support, so the features most relevant to these practices were implemented completely. However, the "sharing space" feature supporting continuous, implicit collaboration within activities was not deployed, in part due to the fact that the participants recruited for the study did not constitute an existing network of close collaborators who would be more likely to use this feature in their everyday work. The amount of time that would have been necessary to fully implement this (relatively) complex feature and to deal gracefully with the associated networking issues (e.g., developing a configuration-free, peer-to-peer desktop synchronization protocol that can operate without requiring changes to participants' firewall settings) would likely have outstripped any potential benefit of including this feature in the deployed version of the software.

6.4 Procedure

Participants were given a demonstration of the features of the Giornata system and then asked to use it for several weeks in the course of their day-to-day work. Participation in the study required a substantive commitment, since I was asking participants to carry out all of their computer-based work for the duration of the study within the context of the Giornata prototype. The average duration of participation in the study was 54 days (max = 82 days; min = 22 days); some participants elected to continue using the system beyond this date; a few are continuing to do so presently (see Figure 6.1).

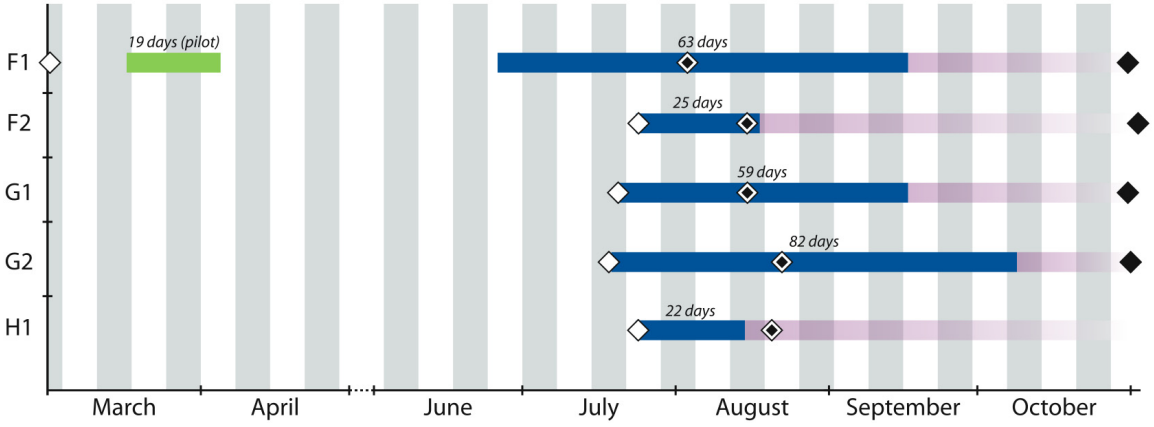


Figure 6.1 An overview of the deployment schedule for the Giornata prototype. Dark blue areas represent dates during which log data of system use were collected, empty diamonds represent dates of initial interviews, partially-filled diamonds represent dates of midpoint interviews, and filled diamonds represent dates of summative interviews.

For the deployment, Giornata was instrumented to log information about when the system was started or terminated, when activities were created or removed, when activity tags were changed, and when switches between activities occurred. At the conclusion of the main portion of the deployment phase (after approximately three weeks of system use), I conducted midpoint semi-structured interviews with each of the participants to

elicit feedback about their experiences using the software and to learn about the ways that Giornata was and was not matching well with their particular work practices (see section B.1). A final set of summative interviews were carried out approximately two months after the conclusion of the main part of the study to elicit feedback about whether the participants had voluntarily continued to use the software or resorted to previous multitasking and task management tools, as well as to probe the ways that Giornata might or might not have affected the longer-term organizational strategies used by the participants (see section B.2). Within the context of these final interviews, I also orally administered surveys comprising a small number of Likert-style questions (e.g., section B.2, questions 6–10).

6.5 Participants

The study participant population initially included ten individuals, recruited using snowball sampling through my academic and industrial research-based social network. The only requirements for participation in the study were that participants have a Macintosh computer running version 10.4.8 or later of the OS X operating system and have the authority to install software on that computer. Participants were not compensated for their participation in the study.

Although all ten participants volunteered for the study, the software could not be installed on two participants' computers due to incompatibilities with their configurations or existing software applications. Three more participants dropped out after experiencing technical difficulties with the Giornata software during the deployment. The remaining five participants completed all portions of the study, except one who was unable to participate in the summative interview, and included two university faculty members (F1 and F2), two graduate students (G1 and G2), and one industrial HCI practitioner (H1). One of these participants (F1) also served as a member of the research project and helped

to pilot the software early in the deployment and vet the interview protocols as they were being developed.

6.6 Results

6.6.1 Overall Impressions of the Giornata System

Based on the Likert-style survey results from the summative interviews, my participants reported having generally positive experiences using the system (see Table 6.1 for a summary). When asked to rate the system on its usefulness using a 5-point Likert scale (1 = "not at all useful," 5 = "very useful"), the average response was 4.2[1]. Many of the concerns expressed about the system's usefulness focused on the stability of the prototype software (which was moderately prone to crashes) and less on the underlying activity-based approach. The participant that gave the system the lowest usefulness value (response = 3.5) explained that he found the activity-based distribution of files and windows useful but the Contact Palette feature of the system less so. He also noted that were it easier to move windows between activities, he would revise his usability assessment to a value of 4.

When asked to rate how well the system allowed them to organize and manage activities fluidly (i.e., without interrupting the way that the participants were working) and how well the system helped them to manage and organize their information (i.e., how well the system's representation of activities matched their mental models of their activities), the responses were also generally positive, with average ratings of 4.2 and 4.0, respectively.

[1] Although the oral survey questions were designed to elicit a numerical rating between 1 and 5 for various aspects of the system, many of the participants improvised and volunteered fractional responses to better convey the degrees of nuance in their impressions of the system.

Table 6.1 Summary of Likert-style survey responses. Participant H1 was unable to participate in the summative interview in which these survey questions were asked.

Likert-style survey question	F1	F2	G1	G2	Average
Usefulness of Giornata (5 = very useful, 1 = not at all useful)	5	4	4	3.5	4.1
Giornata allowed fluid activity management within work (5 = very well, 1 = interruptive)	4.5	3.25	5	4	4.2
Giornata helped organize information in ways that matched mental organization of work (5= very well, 1 = poorly)	5	4	3	4	4.0
Giornata helped in collaborating effectively with colleagues (5 = helpful, 1 = more difficult)	3.75	3	3	4	3.4

The survey question that garnered the least positive feedback asked participants to rate how well the system helped them to collaborate with their colleagues. Although the response indicated that using Giornata was not necessarily a liability—the average response was 3.4, with a rating of 5 indicating that Giornata "helped you to collaborate…effectively within your activities" and 1 indicating that the system "made it more difficult to collaborate"—the lukewarm numerical response and associated comments made it clear that Giornata's collaborative features were among its least well developed. Participant F1 said that although she didn't really take advantage of Giornata's sharing tools, the fact that the system helped to keep her organized implicitly improved her collaboration. Participant G2 took a pragmatic perspective, noting that prior to using Giornata, the best collaboration tool available to him was e-mail and that "Giornata can only improve things."

6.6.2 Logged Use of the Giornata Software

Based on the log data captured by the Giornata system, the participants' usage patterns confirmed at least some degree of success in adopting an activity-centered

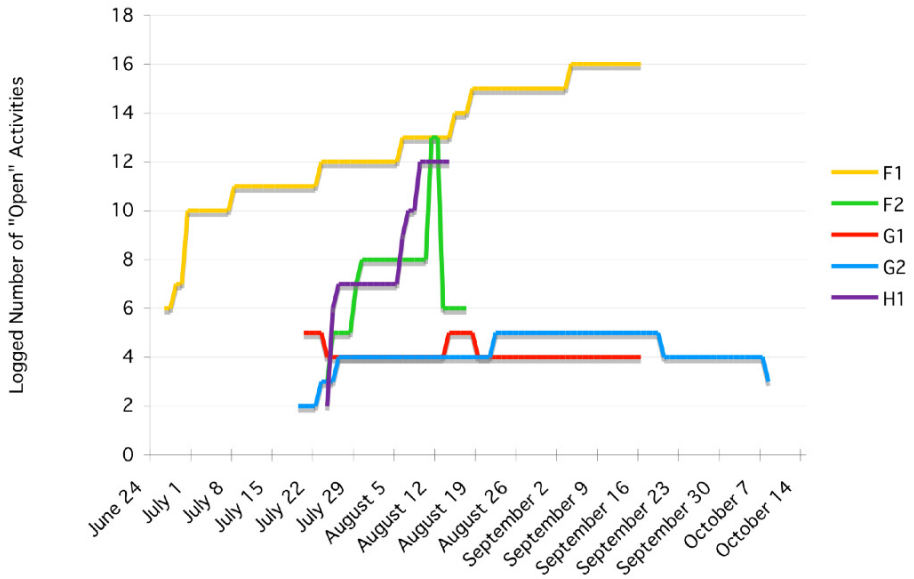


Figure 6.2 The number of "open" activities logged at the end of each day during the Giornata deployment.

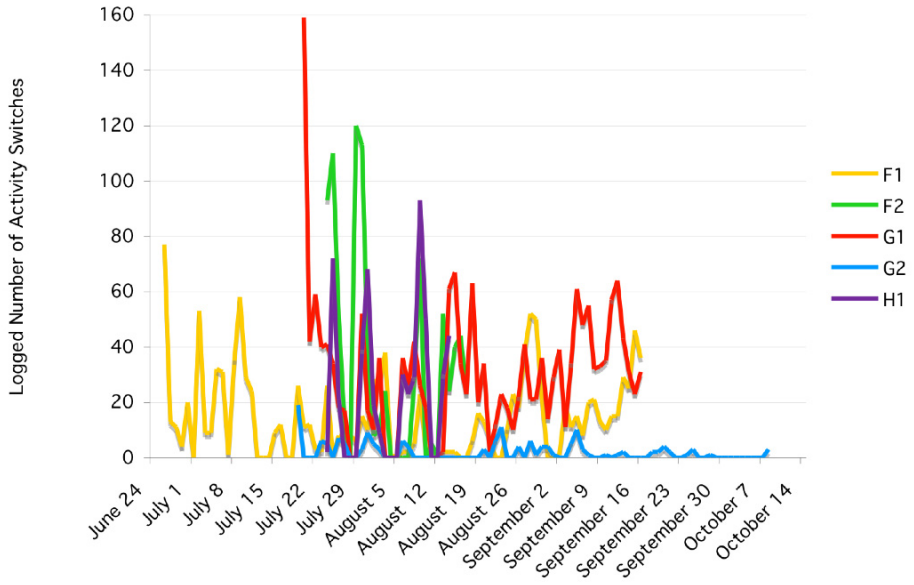


Figure 6.3 The number of activity switches logged on a daily basis during the Giornata deployment. (Sundays during the deployment are indicated by the tick marks on the *x*-axis.)

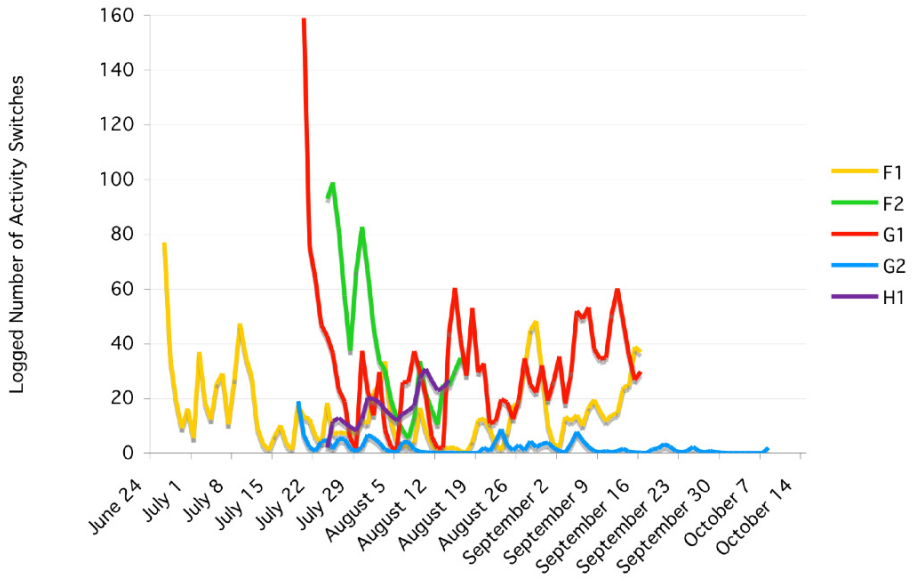


Figure 6.4 A representation of activity switches logged during the Giornata deployment, using single exponential smoothing to reduce local variations and improve the visibility of overall trends in Giornata usage. The smoothing constants used were $\alpha_{F1} = 0.661$, $\alpha_{F2} = 0.350$, $\alpha_{G1} = 0.712$, $\alpha_{G2} = 0.662$, and $\alpha_{H1} = 0.137$; these constants were selected using the least mean square error technique.

desktop model of interaction. The participants maintained an average of 7.6 "open" activities on their systems over the course of the study (Figure 6.2) and switched between activities an average of 28.2 times per day on days that the system was used at all (ranging from participant G2's average of 4.4 switches per day to participant F2's average of 48.2 switches per day; see Figure 6.3 and Figure 6.4).

Some users maintained longer lists of more finely specified activities (e.g., F1, with an average of 13.8 open activities); others created only a few, high-level activities (e.g., G1 and G2, each with an average of 4.2 open activities).

This variability in the granularity of user-specified activities replicates my previous findings (section 2.2) and demonstrates Giornata's flexibility in being adapted to a wide range of work practices. There was no correlation (in either direction) between an

individual's average number of open activities and the average number of activity switches they made each day.

Participants generally tagged activities using one or two words (mean = 1.8 words, standard deviation = 1.122), and, in most cases, assigned these tags to their activities immediately upon creation. Two participants provided at least one tag for every activity they created; the other three participants maintained at least one untagged activity for the entire duration of the study (see Table 6.2). The tags that were used to describe activities initially proved to be quite robust and stable; there were only 12 instances recorded in which tags were added to or removed from an activity over the entire duration of the study (H1 changed activity tags eight times; G2, twice; and F1 and F2, each once). Finally, the participants generally used the tags to label activities descriptively, rather than taxonomically (e.g., used as a list of searchable attributes). Only one participant (F1) re-used any one individual tag to label multiple activities.

Table 6.2 The types of tags used by participants to describe their activities during the Giornata deployment.

Types of Tags Used to Annotate Activities	**Percentage of All Logged Activities Exhibiting One or More Tags of This Type**
A specific project name or identifier	25.9%
The name of an organization or group	16.7%
No tags applied; default value "(untagged)"	14.8%
A conference or event name	13.0%
The name of a software application (excluding "email" and "e-mail")	7.4%
"Email" or "E-mail"	7.4%
A course name or identifier	5.6%
The word "personal"	5.6%
The name of a specific person	3.7%
The name of a specific place	3.7%
A date	3.7%
The words "calendar" or "scheduling"	3.7%

Two participants took advantage of the ability to retroactively apply tags to an activity after it had existed for some time. Participant H1 used of this feature repeatedly, retroactively applying tags to "label" activities associated with nascent projects with their corporate identification numbers once the projects were approved and the numbers assigned.

6.6.3 Interviews with the Study Participants

I interviewed the study participants twice following the deployment of Giornata. The first series of these semi-structured interviews took place approximately 2–3 weeks into the study (see Figure 6.1 for interview scheduling details), and each interview lasted about 30 minutes. The second set of interviews took place approximately two months after the first series of interviews; each lasted about 25 minutes. Except for the interview with participant H1, who lived and worked in another large city across the country and was interviewed using Skype, all interviews took place in person at the participants' workplace. All of the interviews were digitally recorded and manually indexed based on field notes taken during the interviews.

I analyzed the interview data using informal inductive analysis. Quotes from the interviews that were particularly salient—both those that directly addressed the study's research questions and those that stood out as being surprising or interesting—were coded and grouped into thematic clusters.

6.6.3.1 Reflections on Fluid Integration into Practice

The ability to maintain a flexible number of open desktops at a user-definable semantic granularity was cited as a particularly valuable aspect of the system, especially when compared to other virtual desktop implementations:

> I tended to be very lazy when I did [virtual] desktops in the past about keeping them partitioned, which means it became less useful because it was never clear where anything was. And if I'm not paying attention, that can still happen with Giornata, but I think by the notion of binding specific

> activities to specific [desktops] has helped with that…. It may be that there's not that fixed layout [of other virtual desktop managers]. (Participant F2)

Several participants noted that structuring their work into explicit activities provided a valuable, persistent reminder about the state of ongoing tasks. Participants F1 and H1 both discussed ways that their use of Giornata either complemented or served as a substitute for their existing information organization tools (e.g., to-do lists). Participant F1 pointed out one particular to-do function of her activities: she intentionally left several activities open in Giornata solely to serve as reminders for following up with colleagues, even though she knew that she was "not going to do any more work" in these activities. Participant G2 recognized the potential for using activities as reminders, but had difficulties visually distinguishing among open activities; he felt that stronger visual cues would have enabled better ongoing activity awareness.

Several participants spoke of activities as having distinct states, such as "active," "background," and "completed." However, even when they were no longer working in one or more activities, few participants took the concrete step of formally "closing" these activities in Giornata. Participant G1 reported closing several activities (in an attempt to tidy his activity list), only to re-create them a few days later when he realized that he needed to continue work on some aspect of the activities. In the end, he opted to leave all of the activities open that he imagined he might possibly need to return to at a later point in time. This observation highlighted a weakness in Giornata's interaction design: the current implementation of the system makes no distinction between *dormant* activities and those that are *completed*, which raises further research questions about how users conceptualize the different stages in an activity's lifecycle.

Participants also described utilizing a variety of activity switching behaviors, which were often closely linked to the ways that they distributed their applications among their activities. Participant F1 used the first activity created by Giornata as a kind of generic work hub, placing her main e-mail overview window in this activity. When new

e-mail messages arrived, she would open these messages as standalone windows and move them to the most appropriate activity as reminders to engage their contents more deeply or to send a reply. Participant F2 adopted a strategy of opening multiple e-mail overview windows in different activities so that e-mail could always be close at hand, reducing the need to switch back to a single location repeatedly. In contrast, participant G1 intentionally grouped all of his electronic communication tools, including his e-mail application, web browsing windows, and RSS news aggregator, into a single activity so as to reduce the distraction that these tools could create when he was engaged in a focused work activity.

In general, several features of Giornata—particularly the binding of per-activity storage to the desktop and the ability to create and maintain an arbitrary number of activities—enabled the flexible appropriation of Giornata by the study participants. The participants were able to take advantage of the features of Giornata in a breadth of ways that fit in fluidly with their established personal work practices.

6.6.3.2 Reflections on the Encapsulation of Evolving Work

All participants reported using Giornata's activity-based desktop storage. However, the number of files stored within particular activities varied from no items (primarily within communication-oriented activities) to tens of items. Participant F1 said of this per-activity storage that "it feels better than filing," explaining that being able to store files on the desktop caused less anxiety than trying to find the "right" place to put things in the folder hierarchy—an experience she referred to as "soft filing." Participant G2 echoed this sentiment: "I do really like the fact that I have separate desktops and the files go to separate places…I hate navigating through hierarchies, in general…. So, I'd say the file grouping is the biggest win so far." Participant F2 appreciated the fact that the per-activity storage feature actually allowed him to keep *more* items close at hand than he would have previously:

> Actually, having different files on the desktop is a big plus…because I had tended to adopt the approach of trying to keep only one column [of files] on my desktop…because [otherwise] I'd never see them…and that meant that I really didn't have much at all there, whereas that changed a little in the sense that I now can have sort of a column per [activity] and it's less annoying because most of the files on any given desktop are related to that desktop, that activity (Participant F2).

The per-activity desktop storage was perceived by the participants to be so central to Giornata's representation of activities that they often assessed the effectiveness of the system in supporting their activities by commenting about the contents of their desktops. When participant F1 reviewed the desktop contents of several activities during the midpoint interview, she commented there were very few "non-activity" items on any of her desktops, which led her to speculate that the system must have allowed her to create "the right scope of activities."

The participants were quick to identify new information organization strategies that they had developed over the course of using the Giornata system. Participant F1 adopted an approach of moving old project-related folders from her previous desktop structure onto the associated activity desktops: "I think part of what I've started doing was creating more depth in the structures…. So I essentially use that [the archival folder on the desktop] so that it's accessible from my…activity." Participant F2 described a different approach, using the evolving contents of his desktop as a forcing function for *creating* new activities:

> In the end, interestingly, I found that I created contexts based on how much stuff I had on my desktop…. My work pattern became: I would use the desktop in a given context and if I would notice that I had stuff on my desktop that wasn't related to my context, I would move it to the "slough" one at the front—my unnamed one—and then when that started to get a lot of stuff related to one thing, I would create a new context and put it there (Participant F2).

Participant G2 had a pre-existing practice of archiving all of his information to an organizationally maintained server using CVS[2]. As a result, he developed a different interpretation of the per-activity storage as a sort of "temporary holding area" where he kept a duplicate copy of his work-in-progress folder and any ancillary information that he downloaded from the World Wide Web. Whenever he would reach a milestone in his work, he would copy the contents of his work-in-progress folder back to the CVS-controlled folder and then sync it to the server. Although this process required some degree of management overhead, the participant still considered the practice useful: "The stuff that I'm working with at the moment sits on the desktop so I have easy access to it" (Participant G2).

This tension between adopting Giornata's activity-based organization and continuing to take advantage of pre-existing information management strategies was cited as much more of an issue by other participants. Participant G1 had a long-standing practice of storing his files in a particular folder hierarchy outside his normal OS X home directory. For him, the benefits of the per-activity storage simply didn't outweigh his inertia in continuing to manage his content using his established practices: "I don't store stuff on the desktop generally…and that part of my habit didn't change…. If I put anything on the desktop, it's because it's *really* transitory" (Participant G1). Participant H1 relied heavily on the Finder's ability to sort and filter files, a feature that he missed when he tried to adopt Giornata's desktop-centric approach to resource storage:

> I've been storing things on the desktop, but I don't know what I think of that yet. Sometimes, it's really nice that there are some files that are right there, but others…so, I'm looking at [project] right now and I have the [file] version 1.0, 1.1, 1.2, 1.3… they're not really in any order whatsoever on the desktop, which kind of makes finding the latest one a little more challenging…. If it was in a traditional folder structure, it's a little bit easier to do the sorting and that sort of thing… (Participant H1).

[2] CVS, the Concurrent Versions System, is an open-source source code configuration management tool, but it can also be used to create a versioned archive of any digital content. Source: http://www.nongnu.org/cvs/, accessed 25 January 2008.

This participant went on to explain that Giornata's focus on appropriating the desktop as a per-activity store raised additional concerns related to his own personal preferences for organizing information:

> I guess I should also say that I'm one of those people that likes things to be "neat and clean"…but, if I let things go, it's going to reach a very bad state before it gets back to clean. And I fear, when I look at stuff on the desktop, that's the fear I have…if I don't do something, it's going to get really bad before it gets back to good (Participant H1).

When asked about their activity tagging practices, most participants expressed limited enthusiasm for the feature with respect to short-term activity organization. Participant G2's response was representative of most participants' view on using tags:

> I think the tagging *might* be useful. I didn't really use tagging that much for search, but that's probably because my projects are so tight…that it wasn't that useful. But I can imagine when I return to a project at some point…I doubt I'll be able to find everything that I wanted from it, so I imagine that the tagging would be useful in that regard (Participant G2).

Participant G1 echoed this sentiment, generalizing from the short-term utility of tagging to the larger idea of using activities to organize his work:

> I haven't benefited from the payoff…it seems like the lifecycle for the benefit is a bit longer than the two or three weeks that I've been using it, so…it's six months from now when I'm thinking, "Oh, I wrote this paper on this thing, and I know that there's something connected that I have with that…let me go find it." So, within this short period of time, it's hard to know what the benefit is (Participant G1).

Participants seemed to appreciate having a lightweight mechanism for storing content associated with their activities, and cited the visibility of the desktop as a compelling reason to store their resources in this way. However, despite the benefits of this interaction design, several participants voiced concerns about how this new practice conflicted with existing organizational practices and how it made it somewhat more difficult to apply sorting and filtering over information associated with the activity. Participants were also hesitant to pass judgment on the value of tagging their activities, generally anticipating that the real value of this feature might emerge over time.

6.6.3.3 Reflections on Activity-Based Collaboration

The participants reported much less frequent use of the Giornata system's collaboration features than of those features geared towards supporting individual multitasking and information organization. However, even in those cases where participants did not take advantage of the system's collaboration features directly, a common response was that the system still provided implicit benefits for collaboration: "Unfortunately, I didn't get to take advantage of [the Contact Palette], but [Giornata] made me better organized and that helped with collaboration" (Participant F1).

The Contact Palette served as the primary user interface for managing activity-based collaborations. Its interface was specifically designed to take advantage of the existing contents of users' Address Book databases, based on an assumption that because this database was defined to be very flexible and is easily integrated into other applications, Giornata users would likely store most of the relevant contact information using the Address Book application. However, one of the participants in the study had not adopted this practice, relying on her e-mail client and paper-based systems for contact management; another participant used an alternative application, Microsoft Entourage, to manage his contact information. In both cases, the system's heavy reliance upon Address Book integration prevented them from using the system's collaborative features without incurring the additional work of creating Address Book cards on a case-by-case basis.

Participant F2 voiced frustration not with the principle of associating contacts with activities, but with some persistent design issues present in the Contact Palette user interface that made it more difficult to create and maintain these associations. These issues, which were also cited by other participants in the study, included:

- The palette's auto-hide feature, intended to maximize usable screen real estate while keeping the contents of the palette easily accessible, was somewhat annoying because it was easy to accidentally trigger the palette's display, preventing interaction with interface components beneath it.

- Without undertaking the work of assigning custom photographs to each of the contacts in the Address Book database, the Contact Palette simply displayed a column of identical icons to represent the colleagues associated with an activity, making it difficult to identify contacts at a glance.
- Since the pop-up Address Book group in the Contact Palette included every contact in the Address Book database but did not include filter or search mechanisms, finding a desired contact within a large Address Book database to associate with an activity was both error-prone and time consuming.

Even though the close integration with the Address Book and various interface design issues did serve as to deterrents to extensive use of the Contact Palette, several participants did associate colleagues with their activities over the course of the study.

One of the primary reasons participants cited for creating activity–contact associations was to take advantage of the e-mail notification capability that this enabled. One of the unanticipated side effects of these notifications was that they served to heighten some participants' awareness of communicative practices within small groups. Participant G2 commented that his use of the feature revealed the importance of different colleagues' roles in the context of an activity:

> I did put [*colleague names*] in my Contact Palette sometimes, but because…all [these colleagues] are above me in the hierarchy that means that I e-mail them and they don't e-mail me, in general. And, so, the Contact Palette is not so useful. It *was* more useful for the [smaller project], because I had [*colleague*] and [*colleague*] in the Contact Palette…and it did have some utility there (Participant G2).

Other participants felt overly constrained by the way that colleagues had to be associated with each activity in Giornata. Participant F2 expressed a desire to be able to construct groups of colleagues that could be shared from activity to activity and to be able to declare a subset of close colleagues as relevant to all activities. Participant H1 had a particularly succinct way of describing this phenomenon in the context of his work: "I don't have a 1-to-1 mapping between people and activity." He suggested that providing

the capability to link activities with larger social structures, such as e-mail distribution lists, might better represent the more group-oriented ownership of projects that he frequently encountered in his workplace.

Collaboration was clearly one of the most challenging aspects of knowledge work for the Giornata system to support. Participants were able to use the collaboration features of the Giornata system with varying degrees of success, based largely on their existing knowledge work practices and the inertia involved in their use of personal information management tools. Even so, the features were sufficient in some cases to elicit valuable feedback, particularly with regard to organizational communication practices within and across activities and an unexpected phenomenon that can be broadly characterized as "collaboration awareness."

6.7 Discussion

6.7.1 Lessons Learned

The user study revealed a number of interesting findings about how participants adopted (and did not adopt) various features of Giornata's activity-based desktop environment into their day-to-day work practices. Overall, the participants expressed generally positive responses both about both the concept of activity-based computing and the usefulness of this particular instantiation of such a tool. The study also uncovered a significant amount of interplay between the participants' pre-existing knowledge work practices and the activity-oriented approach to information and collaboration management embodied in the Giornata system.

Research Question 1. *Are discrete representations of activities sufficient for representing and supporting the complexity of knowledge work?*

Giornata appeared to be adaptable to a wide variety of working styles, as indicated by the vastly different strategies (e.g., few activities, much switching; many

activities, little switching; few activities, little switching) demonstrated by different participants over the course of the study.

I hypothesized that most activities created in the Giornata system would be transient in nature and represent short-term activities (Hypothesis 1.1). The actual use of the system in large part disproved this hypothesis: over the course of the deployment, only a small percentage of the activities captured were created spontaneously and dismissed in short order; almost none were formally "closed" using the Giornata user interface. This observation raised a number of questions about how participants conceptualize the lifecycle of their activities (including the question of whether a two month deployment is even long enough to be able to distinguish between long-term and short-term activities). One of the concrete outcomes from this aspect of the study was the potential need for activity-based systems to reflect the status of activities at different points along their lifecycle (e.g., *active*, *background*, *dormant*, and *completed*).

I also hypothesized that participants would associate documents with their activities in all but a few cases (Hypothesis 1.2). This hypothesis held true for the vast majority of activities logged by the Giornata system; the number of documents associated with these activities ranged from just a few to tens of documents. However, several participants did create and maintain one or more communication-oriented activities that did not have any documents formally associated with them. In these cases, it is possible that this kind of activity–document association was either not necessary (e.g., for an activity primarily concerned with maintaining a calendar) or because the tools used in that activity maintained their own internal document structure (e.g., iPhoto or e-mail).

Finally, I hypothesized that participants would distribute their "metawork" applications across multiple activities, particularly if those activities were defined at a high level or were active for a long period of time (Hypothesis 1.3). Interviews with the participants reiterated the importance of communication and scheduling tools in most of the activities that they defined. However, participants varied dramatically in the kinds of

strategies they used to "position" these tools with respect to their activities—some attempted to concentrate their metawork tools in a single activity so that they could always be easily found and would present less of a distraction during focused work; others improvised mechanisms for creating instances of or views onto their metawork tools within each of their activities.

Research Question 2. *How does the availability of informal tagging affect resource organization strategies on desktop computers?*

Participants generally appreciated the ability to informally store documents on a per-activity basis; of all of the system's features, this one was perhaps the most well received and actively adopted. Although I did not explicitly measure the amount of time that participants spent filing their documents using Giornata (which would technically have been necessary to verify or disprove Hypothesis 2), participants provided a substantive number of positive comments about not having to traverse hierarchies to accomplish informal filing tasks or expend as much effort as was previously required to find the "right" location to store files.

The only significant concerns that the participants expressed about adopting per-activity information organization resulted from a mismatch between this organizational strategy and their existing information organization practices. The participants that seemed to be the most successful at adopting per-activity filing were those who actively sought out ways to incorporate aspects of their previous strategies into Giornata's "desktop storage" metaphor.

One particularly interesting practice reported during the study was participant F2's strategy of using one unnamed activity as a "slough" space that served as a transitory filing location until a critical mass of related items had been reached, indicating the need to spin off a new activity. This practice of "piling" loosely-related information artifacts resonates very well with previous observations of knowledge work as being

spatial (Malone, 1983; Kidd, 1994) and the practice of more formally filing artifacts as an indication that some meaning for these artifacts has been determined (Kidd, 1994).

Research Question 3. *Is informal tagging sufficient for recording a user's evolving understanding of an activity? Are these tags a useful index for finding relevant resources?*

The study participants most often appropriated Giornata's activity tags to provide static descriptions for the contents of their activities. Contrary to the initial hypothesis that they would add these tags incrementally over the life of the activities (Hypothesis 3.1), there were relatively few changes made to activity tags, except in cases where external influences helped to further refine the original definition of the activity. For the most part, tags were applied to an activity as soon as it was created, and these tags did not change over the duration of the study.

There were, however, a small but significant number of activities—distributed across several participants—that remained untagged throughout the entire study. These "untagged" activities were often used as hubs for "metawork" tools; occasionally, they reflected very short-term activities, although these were far less common than originally predicted in Hypothesis 1.1.

The study findings do not lead to a confirmation or rejection of the hypothesis that tag usage correlates to use of search-based information organization strategies (Hypothesis 3.2). The participants in this study were all relatively light Spotlight search users, which prevented me from comparing tag usage between those who rely heavily on search and those who do not. However, most participants were consistent in their belief that activity tagging provided relatively little value for them in the short term, and that the real value in tagging activities and their associated documents might not be realized until the very long term (e.g., six months or more).

Research Question 4. *Do representations of activity serve as a useful constraint/boundary for managing collaborations?*

The collaboration features of the Giornata software were among the least used for a number of reasons, including the decision to tightly integrate Giornata's Contact Palette with just one personal information management tool and the existence of a number of other user interface flaws in the design of the palette component. Although several participants expressed interest in associating colleagues with their activities (supporting Hypothesis 4), few actually did so during the course of the deployment.

Participants reported that the most compelling reason to associate colleagues with their activities was to take advantage of the e-mail notification functionality built into the Contact Palette. There was, however, no reported use of the Contact Palette to informally share files or to quickly retrieve relevant information about a colleague (in contrast to the latter half of Hypothesis 4); these features may require additional exploration once the barriers to the overall adoption of Giornata's collaboration tools have been addressed.

Finally, some participants expressed concern about having to manually associate individual colleagues with each activity. These participants' feedback that a "1-to-1 correspondence" between activities and colleagues might not be the most accurate representation for some kinds of work environments suggests that it may be necessary to investigate different activity–colleague relationship models and user interfaces for managing them in future iterations of activity-based computing systems.

6.7.2 Relationship of Findings to Prior Empirical Studies and Cognitive Theory

Because Giornata is one of the first systems to provide users with the ability to divide their work into discrete, holistically defined activity clusters, the findings from this study serve to illustrate how users might adopt and appropriate this class of activity-based technologies. The findings can additionally be a useful tool for reflecting back upon previous characterizations of knowledge work and the theory that has been put forward to describe the structure and content of the activities that serve to define and guide these professionals' work.

The empirical studies carried out by Kidd (1994) and Barreau and Nardi (1995) resulted in the identification of several characteristics of knowledge work, many of which drove aspects of the Giornata system's design. By examining how the participants in the deployment and user study responded to these aspects of the completed system, the claims can be assessed from a new perspective.

6.7.2.1 Knowledge Workers' Low Dependence on Filed Information

> *"Knowledge workers, in general, have a low dependence on filed information" (Kidd, 1994).*
>
> *Desktop computer users express "[a] 'lack of importance' of archiving files" (Barreau & Nardi, 1995, paraphrased by Freeman & Gelernter, 2007).*

Participants in the Giornata user study discussed their use of the system almost exclusively in terms of the artifacts that the system allowed them to keep "at hand." Those participants who reported having formal processes for moving content from activities into long-term archives (e.g., participant G2's use of CVS) talked about taking these steps primarily at activity milestones and primarily for the purpose of having a back-up copy of the information stored on a file server. These findings help to confirm the primacy—at least on a day-to-day basis—of information resources immediately relevant to ongoing activities over those artifacts that have already been classified and archived.

Furthermore, the study participants "closed" activities very rarely over the course of the deployment. Instead, most participants kept their activities open, either leaving their content in whatever state that it had been in while working on it (implying that the "working state" of the per-activity storage was "good enough" for the long term) or simply re-conceptualizing the objective of the computer's activity representation to reflect a transition to a new, but related task.

A potentially interesting follow up study would be to examine how files are managed when these kinds of systems are used over the very long term: would users be

more likely to archive documents associated with activities within the activities themselves or to intentionally move artifacts into other, more structured file hierarchies when activities are brought to a close?

6.7.2.2 Knowledge Workers' Reliance on Spatial Layout as a "Holding Pattern"

> *"The spatial layout of a knowledge worker's materials is important as a 'holding pattern' for short-term organizational purposes and before the materials have been classified and can be filed" (Kidd, 1994).*
>
> *Desktop computer users express "a preference for location-based search for finding files (in contrast to logical, text-based search)" (Barreau & Nardi, 1995, paraphrased by Freeman & Gelernter, 2007).*

The participants in the study strongly validated these claims about knowledge workers; they almost unanimously cited the informal per-activity storage as the most compelling feature of the Giornata system. Participants talked about how placing items on the desktop felt "better than filing" (Participant F1), that the system's design resonated will with existing practices of storing "temporary" or "working" versions of files on the desktop, and that having a place to collect all of the items related to an activity significantly strengthened the relationship between traditional realizations of virtual workspaces. The relatively common use of untagged activities as an unstructured "holding pattern" for materials that did not yet have a formalized purpose or structure also supported this assertion.

One observation from the study that is not emphasized strongly in previous characterizations of knowledge work is the perceived importance of being able to manipulate these short-term organizational materials in a variety of different ways in order to make sense of the artifacts associated with an activity. Participant H1's comments about the importance of *sorting* and *filtering* capabilities of traditional Finder windows for finding files and tracking their evolution through multiple versions suggests that spatial layout, itself, may not be sufficient for managing the increasing number of digital artifacts that knowledge workers bring to bear on some of their activities.

6.7.2.3 Knowledge Workers' Reliance on Spatial Layout as a Primitive Language

> *"The spatial layout of a knowledge worker's materials is important as a primitive language, since the physical (and, presumably, digital) artifacts stand in as a model of real-world phenomena" (Kidd, 1994).*

A significant percentage of the activities defined using Giornata were tagged with project names, organizations, or events—the same identifiers used to describe ongoing conceptual units of work prior to the deployment of the Giornata system. One of the participants who had previously used (and subsequently rejected) other virtual desktop software also speculated that he had been more successful using Giornata because it provided a better mapping between real-world activities and virtual desktops than did other systems. These observations support the claim that digital artifacts stand in as a model for real-world conceptualizations of activity and that the closer this mapping can be, the more likely the system might be to succeed.

6.7.2.4 Knowledge Workers' Reliance on Spatial Layout as a Reminder

> *"The spatial layout of a knowledge worker's materials is important as a contextual cue for resuming a suspended activity, the location of artifacts helps to answer the question, 'where was I?'" (Kidd, 1994).*
>
> *Desktop computer users cite "the use of file placement as a critical reminding function" (Barreau & Nardi, 1995, paraphrased by Freeman & Gelernter, 2007).*
>
> *Miyata and Norman (1986) claim that spatial location is a kind of memory aid without need for cognitive processing of icons or text.*

The study confirmed many long-standing assertions that representations of activity are a powerful tool for reflecting over the landscape of activities currently under way and reminding knowledge workers about work that still needs to be done. Participants in the study talked very specifically about using Giornata's open activity list in lieu of to-do lists and several participants made suggestions about how the system's visual representation of ongoing activities could be strengthened to improve its usefulness as an "at-a-glance" tool for assessing the state of all open activities.

6.7.2.5 Knowledge Workers' Reliance on Spatial Layout as Demonstrable Output

> *"The spatial layout of a knowledge worker's materials is important as demonstrable output, since piles in some ways quantify the work that has been accomplished" (Kidd, 1994).*

During the semi-structured interviews, participants frequently referred back to their activity lists as evidence of the progress they had made in better organizing their ongoing activities and the progress that had been made towards accomplishing their activities' objectives. The language used by many of the participants when talking about their ongoing activities suggested that they evaluated their success in completing an activity by whether they felt that they could "check it off," that, in essence, the fewer activities they had open in Giornata, the less outstanding work there was left to do.

6.7.2.6 The Importance of the Individually-Focused, Discrete Activity as Described by Activity Theory

The design of the Giornata system was also influenced heavily by cognitive representations of activity, especially Activity Theory (Halverson, 2001; Leont'ev, 1978; Vygotsky & Cole, 1978, Engeström, 1987; Boer, van Baalen & Kumar, 2002; Kaptelinin & Nardi, 2006). This theory suggested that an activity-based system should provide a series of discrete activities that each incorporate the diversity of tools used to accomplish that activity as well as representations of the social context within which the activity takes place.

While the computational representations of activity inspired by this model were praised by the participants for unifying many of the relevant aspects of an activity in a single user interface, several tensions were also uncovered. Participants appreciated the fact that Giornata allowed them to separate their work into distinct activities, enabling them to keep more of the "relevant" resources for their work at hand at any given time. However, use of the system revealed an unexpected level of interconnectedness among the activities that these users defined. Not only were information artifacts (e.g.,

documents and e-mails) shared across multiple activities, but several participants pointed out a frequent need to access instances of running applications across multiple contexts as well. This apparent contradiction begs a number of questions: Do the "soft boundaries" that participants reported around their activities suggest that the ad hoc definition of activity structures is inherently error-prone? Can activities actually be represented as distinct clusters of tools and artifacts in practice? Are an individual's "real-world" activities so fluid and interdependent that computational representations need to encode the explicit relationships among activities in addition to the structure within the activities?

Giornata also provided lightweight support for representing the social context of individual activities on an activity-by-activity basis. However, several participants provided feedback that maintaining representations of social groups within activities was difficult and time-consuming, commenting that the individuals and groups that they associated with one activity were likely to be relevant collaborators within other activity contexts at the same time. One of the study participants (H1) drew particular attention to this problem, suggesting that instead of activities *containing* a social context, it would be more useful in his work environment to define activities from *within* a social context; that is, given an existing team in the workplace, a better approach might allow activities to be created based on the existing composition and shared resources of that team. At a minimum, he—and other participants—pointed out that it should be possible to more readily share representations of pre-existing social structures across multiple activities.

In general, the study revealed that although activity theory might work well as an analytic tool for *understanding* activity, applying these models as a framework for *organizing* work in the real world requires significant effort to appropriately support "tool" re-use across multiple activities, to represent the sometimes-complex interrelationships among activities, and to provide interfaces that allow appropriate

association of activities and social groups (or vice versa), even from the perspective of a single activity-based system user.

CHAPTER 7

TECHNICAL INFRASTRUCTURE AND THE DEVELOPMENT OF ACTIVITY-BASED COMPUTING SYSTEMS

Giornata is one of a relatively small number of fully implemented systems that both provide comprehensive support for activities and complement existing knowledge work tools. The process of designing and implementing this system yielded a variety of technically-oriented insights, ranging from observations about how state-of-the-art operating systems and user interface toolkits sometimes facilitate—and sometimes hinder—the development of activity-based tools to critiques of the limitations of established information organization principles and user interaction paradigms. In this chapter, I present these insights, together with lessons learned during the design and implementation of Kimura and the sharing palette, to provide a broad summary of technical requirements for supporting activity-based computing systems.

Each of the three systems that I have described in this research was developed on a unique computing platform and designed to meet a particular set of implementation criteria. Taken together, these diverse experiences can be used to reflect on the relative suitability of the technical characteristics of these different platforms in supporting the design, implementation, and deployment of activity-based computing systems. Kimura was written in a combination of Java code and native C++ libraries, was executed on the Windows 2000 operating system, and incorporated distributed programming techniques to accomplish cross-machine display coordination and data persistence. The sharing palette was implemented in Java and took advantage of a combination of cross-platform frameworks (e.g. Swing) to define the user interface and native libraries (e.g., Apple's Bonjour implementation of the multicast DNS protocol) to support local discovery and file sharing. Giornata was implemented using a combination of C, Objective-C, and

AppleScript[1] code on Mac OSX, integrating closely with a variety of application programming frameworks, including Carbon, Cocoa, and application-specific programming interfaces for components like Address Book, Mail, and Spotlight[2]. (The particular implementation details are discussed at length in section 5.3.) Each of these implementation experiences exposed design decisions embedded in the respective platforms that sometimes aided research and development in activity-based computing and sometimes dramatically limited what could be built and how the interaction would play out from the perspective of a potential user.

Because representations of activity are in some ways orthogonal to the existing application- and document-centric model of computing espoused by traditional operating systems (e.g., one activity frequently spans multiple applications and draws on multiple kinds of information resources, whereas the application-centric model gives primacy to a single application and document within the context of a particular task), the development of activity-based systems, particularly those that work alongside existing applications, presents a number of distinct technical challenges. Although it would certainly be possible to completely re-engineer an operating system to foreground activity at the lowest levels, building such a system from the "ground up" with sufficient robustness and functionality to support long-term deployment and evaluation of the system's features would be prohibitively difficult in the context of academic research. The alternative approach—and the approach taken in the course of this research—has been to selectively augment existing systems with activity-oriented tools that work as closely as possible within established interaction metaphors and build upon conventional programming interfaces and toolkits.

This chapter examines three of these operating system components—the window manager, the filesystem, and application development and runtime tools—that have

[1] http://www.apple.com/applescript/, accessed 24 January 2008
[2] http://developer.apple.com/macosx/spotlight.html, accessed 20 January 2008

served particularly important roles in the research systems described in this dissertation. For each of these components, I discuss ways that different design decisions within the operating system helped both to foster development of activity-based computing tools and to complicate or hinder the implementation of prototype systems in this domain. It is my hope that this chapter will serve as a valuable resource to foster future research in activity-based computing and to suggest new directions for research in other, traditional domains within computer science.

7.1 The Window Manager

The concept of overlapping windows was developed as a key component of the graphical user interface, enabling computer users to simultaneously manage multiple interaction contexts. Although the graphical window manager is widely recognized as the technology responsible for enabling the rise of computer-based multitasking, studies emerged relatively shortly after the widespread adoption of the technology pointing out its inherent weaknesses in the face of "real-world" computer use (e.g., Bannon, Cypher, Greenspan & Monty, 1983). One of the most significant shortcomings of traditional window managers is the fact that they typically incorporate little or no support for managing clusters of windows that, together, constitute the applications and documents used to accomplish a single, coherent task. As a result, the window manager is a core operating system component often manipulated, if not augmented or replaced outright, in the context of activity-based computing systems.

However, enhancing, augmenting, or replacing the window manager is rarely a straightforward proposition. Window managers are also typically designed to shield application developers from needing to deal with the fine-grained details of how windows are rendered and managed on screen. There are also pragmatic issues of platform security at play, in that applications with "deep hooks" into window management functions have the ability to effectively hijack the entire computer display, close or disable other

application windows, or even (potentially) snoop on private data by intercepting interaction events destined for other applications' windows. (Window management and event distribution functions are often tightly coupled.) As a result, it is not unusual for the inner workings of the window manager to be quite tightly encapsulated within the lower levels of the operating system and to be made relatively inaccessible to application developers.

This situation presents an inherent tension in developing activity-based computing systems: few operating systems have any substantive built-in support for managing windows in semantically meaningful clusters and when they do, the ability to query or control the grouping mechanism is typically quite limited. On the other hand, developing activity-based tools that can exert control over the operating system's window management polices is inherently difficult, since access to window management functions through the application development framework is also restricted.

7.1.1 Window Management and Fluid Work Practices

An increasing number of mainstream operating systems are incorporating support for managing windows in a virtual desktop or multi-workspace environment. While this capability has long been included as a default component of X Windows-based window managers, it has only recently been introduced to general computer users as a pre-installed component of a widely-used mainstream operating system such as "Spaces" in Apple's OS X version 10.5 ("Leopard"). While several third-party utilities are available to enable this functionality in Microsoft Windows, this capability is still not exposed to end users in off-the-shelf configurations of the operating system.

That virtual desktop managers are included in or omitted from different operating systems is less an issue for activity-based computing research and development than is the availability of infrastructure at the level of the window manager to support cross-application manipulation of windows. As long as some capability for enumerating and

controlling the visibility of application windows is available, it becomes possible to—at a bare minimum—*simulate* virtual desktop behavior, even when no formal capabilities for grouping windows exists in the operating system, itself. This is the approach that I took in developing Kimura on Windows: in order to create the illusion of multiple workspaces, the Kimura application manually monitored other applications' requests to create and destroy windows and then manually toggled the visibility of these windows when the user switched between open activities. Although this worked in most cases, individual applications did not have any awareness about whether they were currently visible or not. As a result, it was relatively common for application dialogs, for example, to appear in incorrect activities or for the system to occasionally "lose track" of to which activity a particular window belonged. One positive aspect of this approach on Windows is that hiding an application's window effectively renders it invisible in all aspects of the user interface, including the Taskbar. This allows the Taskbar to accurately reflect all of the windows currently associated with the active virtual desktop. Unfortunately, this does not extend to the quick application switcher that is invoked by pressing Alt + Tab, which results in the display of a very long list of all applications open across all ongoing activities.

Although the last several versions of the OS X window manager have included internal support for grouping windows into virtual desktop-like "workspaces," no user interface-level support for accessing this functionality or managing these window groupings has been available until the most recent release of the operating system (version 10.5, "Leopard"). The presence of this infrastructure reduced the amount of code required to implement Giornata, although the lack of formal documentation for interacting with the API and the absence of any built-in user interface components for obtaining an overview of all populated virtual desktops or moving windows between virtual desktops did complicate the development effort. Furthermore, because Apple's engineers likely did not anticipate that third-party developers would appropriate this

undocumented infrastructure, they restricted the execution of certain API calls to the process hosting the window manager on the system. (This decision was likely also made in order to optimize certain resource-intensive window management interactions, such as Exposé.) As a result, it became necessary to overcome additional hurdles for extending the functionality of the default window manager, including the use of Mach code injection, as described previously in section 5.3.1.

An interesting trade-off between these two techniques came to light during the deployment of the Giornata system, when several of the study participants asked whether it might be possible for a single window to exist in multiple (but not necessarily all) open activities. This kind of functionality is easy to prototype when the activity-based application is managing all of the window-to-virtual desktop mappings manually, but becomes much more difficult when building upon existing frameworks that might enforce constraints such as requiring windows to be associated with one and only one virtual desktop or "workspace" at a time. This type of trade-off exemplifies a common tension in working with such fundamental building blocks of the operating system, which often seek to balance providing programmers flexibility in the kinds of capabilities they can provide with offering users a more concrete sense of security that rogue applications will not be able to modify low-level aspects of the system in undesirable ways.

7.1.2 Window Management and Encapsulating Evolving Work

One of the primary reasons that Giornata's participants requested the ability to "share" a window across multiple open activities is due to the fact that some applications are designed to provide a single-window portal to information resources that either cannot be manipulated individually or that make more sense to manage in the aggregate; e-mail, virtual machine hosts, and media libraries are among the most common examples. These applications already serve to encapsulate similar *types* of information in a single window, but different parts of this information space are likely to be more or less relevant

when working in a given activity (Bergman, Beyth-Marom & Nachmias, 2006). These single-window portal applications are perhaps the most appropriate approach when the "work" that they support focuses on viewing or organizing collections of resources that are saved for reference or consumption but generally not edited (e.g., e-mail messages, photos, songs, web pages). Although incorporating an awareness of the current activity into these tools could certainly provide a host of benefits for quickly finding and managing those items most salient to a particular task, doing so would almost invariably require the application itself to be re-written to take advantage of the user's activity context.

In contrast, many window systems (and application development environments) have begun to evolve in ways that discourage this kind of single-window application design when providing generative tools for creating or editing information resources whose contents are more dynamic. The general trend toward representing each of these documents (in the broad sense) within its own top-level window has been helpful from the perspective of successfully introducing activity-based organization into an existing application ecosystem. During the earliest attempts to realize the Kimura system, for example, the contemporary version of Microsoft Word used a single-window interface by default, making it impossible for the system to distribute word processor documents of the same type among multiple activities. The one-document-per-window model that has become more common on Windows in recent years (and has been the default application design paradigm on the Macintosh platform since its inception) has dramatically simplified the process of managing these documents at a semantic level, together with supporting information resources of other types.

Should activity-based computing gain traction as a mainstream interaction paradigm, it might be useful to either re-examine the design of single-window applications to respond to changes in external context, such as activity switches, or to expose a semantically-oriented representation of their content to the underlying window

system so that activity-aware tools can exert control over and filter the contents of an application's views and the tools it makes available at a given time.

7.1.3 Window Management and Collaboration

Although collaborating around document instances is a widespread practice among knowledge workers, it is much less common for colleagues to collaborate over a live document except when collocated, as tools for doing so are still relatively scarce. As a result, the benefits of augmenting a window manager with activity-based capabilities translate only indirectly to increased collaboration support, insofar as awareness information relevant to the current activity might be made more visible and less likely to be lost in a sea of background "noise."

Mainstream operating systems are increasingly providing built-in support for sharing one's entire screen with colleagues, such as Microsoft's Remote Desktop Connection[3] and OS X's Instant Screen Sharing[4] features, both of which build on previous research systems like VNC (Richardson, Stafford-Fraser, Wood & Hopper, 1998). These tools can be used in the context of particular activities in order to create tightly-coupled working environments as needed; however, these applications suffer the same drawbacks as the single-window applications discussed in the previous sections. In addition, they are typically built to transport the entirety of a user's display across the network, regardless of the relevance of the information displayed on different parts of the screen. Further work in combining activity-awareness with limited-scale screen clipping applications (e.g., Tan, Meyers & Czerwinski, 2004) might provide one means for enhancing collaboration on an activity-by-activity basis at the level of the window manager.

[3] http://www.microsoft.com/windows/products/windowsvista/features/details/remotedesktopconnection.mspx, accessed 7 May 2008.

[4] http://www.apple.com/macosx/features/300.html#finder, accessed 7 May 2008.

7.1.4 Summary of Technical Requirements for Window Managers to Better Support Activity-Based Computing

In order to better support the development of activity-based computing systems, future window management systems should:

- allow users to cluster windows into groups and provide interaction techniques that are appropriate for managing those groups;
- allow individual windows to be shared across multiple—some or all—virtual desktops or window groups;
- provide capabilities for third-party applications to customize the system's window management strategies and user interactions, at least for some class of privileged system utilities (e.g., activity managers with elevated execution privileges);
- issue system-wide notifications about changes to the window manager configuration (e.g., the receipt of an activity change request) that applications—particularly those composed of a single-window—can use to filter the information they present or to modify the display strategies they employ; and
- enable collaboration applications to more easily share portions of the screen or the contents of individual windows with remote collaborators without requiring manual capture-and-redirection of those windows' content.

7.2 The Filesystem

As the computer's long-term information repository, the filesystem used by an operating system dramatically influences how a system's users manage, organize, store, and retrieve information resources in the context of their work. Historically, mainstream operating systems' filesystems have adhered to a hierarchical filing strategy, requiring that all information be stored in a single, tree-like structure on each logical volume

(usually, a disk partition) and that each file be stored in a single location within the tree, uniquely identified by an alphanumeric filename. Although it is possible to create more complex structures in many modern filesystems (e.g., files whose contents are stored in one location but appear to exist simultaneously in other locations on the disk via "links" or "shortcuts"), the lack of discussion on the topic in the relevant research literature and feedback from our study participants suggest that most users are not willing to invest the time to maintain complex, cross-linked organizational hierarchies for their files.

Because a significant component of activity-based organization is determined by and reflected in the mechanisms used to classify, sort, and file information resources, the capabilities and limitations imposed by the design of the filesystem can have a significant effect on how well systems are able to provide support for activity-based computing. Even without explicit tools in place to provide activity support at the level of the filesystem, many users already appropriate the structure and naming conventions of their folder hierarchies to provide scaffolding for managing their information resources into semantically meaningful groups (Barreau & Nardi, 1995).

7.2.1 Filesystems and Fluid Work Practices

In many filesystems, a file's identity is established based on the *path* to the file: a description of the sequence of folders that must be traversed to find the file, followed by a name that uniquely distinguishes the file from others stored within the same enclosing folder. This organization scheme, while common, clear, and relatively straightforward does have its drawbacks, among them several assumptions that exist in direct conflict with studies of the ways that knowledge workers mentally conceptualize their information resources (e.g., Kidd, 1994). One of the most significant shortcomings of the hierarchical filing scheme is the fact that the system requires users to *find a location* for all files upon creation and to provide them with a *unique name within that location*, often before any significant work has been done in the file that would help to inform this

entirely front-loaded meaning-making process. In response, Giornata was designed to ameliorate one of these two requirements: rather than forcing its users to find an appropriate place to store a file, any information resources created in the context of an activity could simply be stored on the desktop, itself. The contents of the desktop are automatically—and semantically—bound to the current activity in an extremely lightweight manner.

The implementation for this per-activity storage feature required that Giornata be able to dynamically swap the contents of the desktop in and out as the user transitioned among different activities. This would be a trivial process if there were a guarantee that files would only be read from or written to when the associated activity were active and the files could always be found on the desktop; however, in reality, these files needed to be moved elsewhere when an activity was relegated to the background. If Giornata had been developed on a platform whose application development model encouraged regular path-based references to files (e.g., Microsoft Windows), this would have been a significant roadblock to implementing the feature. However, another aspect of the underlying filesystem used in OS X—handle-based file input and output—provided an elegant and robust solution.

When an application opens a file for reading and writing in a UNIX-based environment (e.g., OS X), the operating system provides the application with a handle representing the file, which becomes a token for further file accesses. These handles remain active and valid even if the underlying file is renamed or moved within the filesystem. This approach provides applications with a robust mechanism for indirectly and consistently manipulating files. In the case of Giornata, handle-based file input and output also enabled the realization of a per-activity storage feature, an aspect of the design that appealed most broadly to the participants during the deployment.

In Giornata, the per-activity storage was implemented using brute-force management of the user's desktop items by the system. When the user initiated a switch

between two activities, Giornata invoked a short AppleScript function that performed four functions:

1. The script recorded the (X, Y) position of each file stored on the desktop in a hash table associated with the outgoing activity.
2. The script asked the Finder—OS X's file management application—to move each of the items in the user's desktop folder into a folder associated with the outgoing activity. Making this call through the Finder, as opposed to using the standard C file management APIs, ensured that all extended file information—including the finder comments in which the activity tags were stored—were correctly moved along with the file's contents.
3. The script asked the Finder to move each of the items in the folder associated with the incoming activity to the user's desktop folder.
4. The script traversed the hash table associated with the activity and re-positioned each item found to its previous (X, Y) location on the desktop surface.

Because running applications continued to hold valid file handles for these files regardless of their location in the filesystem (the user's desktop or the folder associated with an inactive activity), the running applications continued to function normally without modification, even when they held references to documents opened in multiple activities simultaneously. The only drawback to this approach was that a few applications' "most-recently used" file lists ceased to function correctly when *unopened* files last known by the application to exist on the desktop were moved elsewhere as a side-effect of a Giornata activity switch; however, no participants identified this as being a particularly critical problem at any time during the system's deployment. In contrast, developing Giornata on another platform where there is much more reliance on path-based file access at either the operating system or application toolkit layer would have

necessitated implementing a much more complex solution in order to support per-activity storage on the desktop.

7.2.2 Filesystems and Encapsulating Evolving Work

The hierarchical organization approach works well for describing the contents of archival information stores when the contents are assumed to be relatively well defined and the relationship among artifacts clearly established. However, in the early stages of an activity, when the relationships among artifacts are less well defined and the identity of and eventual use for each individual information resource may be in flux, this kind of path-oriented structure becomes less reliable, since the names and locations of resources may change frequently. Because tools like the sharing palette are designed to support collaborations that take place around the individual information resources most relevant to a shared activity, it is necessary that changes to the shared files are immediately recognized and propagated to all of the stakeholders in the collaboration. However, this kind of tight monitor-update capability presupposes that the file, itself, is an established and moderately stable boundary object around which the collaboration can take place. Files that are constantly moved and renamed to reflect their intended role and relationship to other files are much more difficult to track and update using this kind of file monitoring infrastructure.

Another possible solution to the highly dynamic nature of information resources generated and revised over the course of an evolving activity is to revisit the underlying naming mechanisms provided by the file system to provide more flexible tag-oriented file clustering and naming, as opposed to a traditional, hierarchical representation (after Dourish et al., 2000). Because the use of informal tagging has been received with some degree of enthusiasm at the activity level with the Giornata prototype, applying this organizational strategy to the information resources, themselves, might enable users to create new kinds of informal and emergent, cross-activity organization schemes for

storing their artifacts. However, providing this kind of functionality would require substantial redesign of the filesystem layer and potentially necessitate the modification of individual applications to support this new kind of organizational scheme.

A separate but related problem is reflected in the sheer volume of information that knowledge workers interact with on a daily basis. Search-based strategies are an essential tool for navigating the increasing amount of information available online and arriving in e-mail inboxes on a daily basis. As the use of search tools becomes more prominent on the Internet, a parallel movement is playing out on the desktop as an alternate means for organizing information stored locally on a computer's hard disk. Because the cost of purchasing digital storage is so low, the conventional wisdom is that all digital artifacts can be archived indefinitely. As a result, the act of finding a previously stored document can be quite costly. The movement towards search-based information management (at least initially) reduces much of this cost: store the document anywhere on the disk and it is still possible to find it instantly, so long as sufficient context about the activity in which the file was used and/or metadata about the file's contents have been indexed to make it possible to uniquely locate the file in the proverbial haystack.

Desktop search tools like Windows Search[5] and OS X's Spotlight framework rely on collecting and indexing a rich stream of metadata about all of the files stored locally in order to provide relevant responses to users' queries. This dependency poses two main issues for filesystem developers: (1) filesystems need to incorporate an increasing number of tools for identifying or extracting relevant metadata to apply to files, and (2) filesystems need to support exhaustive indexing of this metadata to facilitate the actual search requests. Activity-based systems can help to address the first of these two issues by serving as a source of contextual metadata at little or no cost to the system's user;

[5] http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx, accessed 7 May 2008.

Giornata demonstrates one mechanism for providing this kind of semantically meaningful annotation to files used over the course of tagged activities.

7.2.3 Filesystems and Collaboration

Giornata and the sharing palette both demonstrated user interfaces that were designed to simplify as much as possible the process of collaborating over digital artifacts. A significant portion of the development work for both of these systems involved code that would monitor and synchronize files across two (or more) collaborators' personal filesystems. While each of these systems used separate mechanisms for detecting changes to shared files—the sharing palette, by examining shared files' sizes and "last modified" dates once every second, and Giornata, by subscribing to a distributed notification system to receive callbacks whenever file changes were detected anywhere in the filesystem—both shared a custom peer-to-peer protocol for keeping all collaborators' copies of the files up-to-date.

Although these "shared filesystem" implementations provided enough functionality to demonstrate the respective user interface designs of these two systems, they also served to identify a number of persistent challenges that need to be addressed in this domain before a larger adoption of these collaborative file sharing interactions can take place. These challenges include questions about how to correctly handle simultaneous editing of a shared file by multiple collaborators, how the particulars of access control should be handled (particularly with respect to what happens when sharing permissions are revoked and how—or if—redistribution of shared files should be managed), and where files that are shared should be stored in order to ensure access to shared materials when one or more of the collaborators disconnects from the network.

7.2.4 Summary of Technical Requirements for Filesystems to Better Support Activity-Based Computing

In order to better support the development of activity-based computing systems, future filesystems should:

- incorporate mechanisms for annotating individual files with metadata describing who used or modified the files, through what application(s), and in which activity contexts;
- allow open files to be moved, renamed, or annotated without breaking applications' existing links to the files;
- contain tools that can allow users to search for files based not only on filename and content, but also on the context in which the files were used or, perhaps, based on what other files were in use at the same time;
- enable files to be stored in informal "piles" (e.g., unnamed clusters) without requiring that they initially be assigned a formal (or final) filename;
- provide robust mechanisms for concurrent modification of a file by multiple users or, at a minimum, built-in facilities for managing and synchronizing multiple versions of a file; and
- offer services for seamlessly combining local and shared filesystems to simplify the process of sharing files among collaborative groups, guaranteeing the consistent availability of files to all parties even when one or more file providers or recipients becomes disconnected from the network.

7.3 Application Development and Runtime Tools

It would be infeasible to prototype an activity-based computing system were it necessary to implement both the novel, activity-based aspects of the system as well as to re-implement the multiplicity of tools that knowledge workers rely upon, in the interest of creating a coherent, consistent, and deployable system. However, building activity-based

tools that operate in isolation would also be of limited use; it would be virtually impossible to learn anything substantive about users' perceptions of and practices for organizing their activities without allowing them to do so in the context of their existing applications and data.

7.3.1 Applications and Fluid Work Practices

The development of activity-based tools has become much more practical over time, given the increased stability of recent desktop operating systems and the falling cost of equipping personal computers with enough random-access memory to support concurrent execution of many applications spanning a variety of ongoing activities.

Just as operating system-based multitasking enabled users to utilize multiple applications simultaneously in the course of one activity—or, at most, a few activities—the availability of ample memory and multicore processors enables simultaneous work in many activities, each drawing on multiple applications and documents. During the development of the Kimura system, the underlying operating system architecture (Windows 2000, at the time) was rarely stable enough and the available desktop computer hardware had so little memory available (typically less than 512Mb) that keeping more than a few activities "open" and running over the course of even a scripted demonstration was a significant challenge. In contrast, the development of Giornata on Mac OS X, a UNIX-based platform designed specifically for multi-user and multi-process environments, and running on stock laptop computers with 2 or more gigabytes of memory has enabled long-term deployment and use of the activity-based software in real-world contexts, supporting up to 15 concurrent activities with comparatively few technical problems. In fact, none of the issues that participants reported when using Giornata involved problems with low system resources or an inability to work in as many activities or applications as desired. (There were some concerns expressed about how large numbers of activities were *represented* in the user interface, particularly in the

quick activity switcher, but these problems could easily be rectified with a minor interface re-design and the use of a more appropriate tag rendering strategy.)

It has long been recognized that one of the fundamental limitations in developing applications that manage groups of other applications on the system is that there are no obvious mechanisms for restoring the state of the controlled applications should the system crash or be restarted (Robertson et al., 2000; MacIntyre et al. 2001). The ideal scenario—and an often-requested feature—would be for systems like Task Gallery, Kimura, or Giornata to be able to return the user to a previous activity at precisely the point that they last left it, with the same applications open, displaying the same documents, with windows stacked and positioned in the same configuration, views scrolled to display the same content, and the same interaction mode active. Accomplishing this kind of activity recovery is impossible given current application programming paradigms, as it is neither feasible to ask an application to describe its execution state at this level of detail, nor is it possible to instruct applications to assume a particular execution state, even if that information were available. Currently, the best that most activity-based systems can do is to preserve the arrangement of windows in an activity while those windows remain open, and—with some degree of effort—restart applications that were running and open documents that were loaded the last time the activity was visited.

There are two ways that this kind of detailed application state management could be realized in future systems. The first approach would be to change the way that applications are developed, requiring that application developers implement a pair of standard functions that the operating system can call to store the current execution state or load a previously saved state. A second approach would be to implement an application runtime environment based on the concept of virtualization:

> Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning,

> time-sharing, partial or complete machine simulation, emulation, quality of service, and many others (Singh, 2008).

Virtualization can be used to run multiple operating systems side-by-side on a single computer by providing an abstraction for each of the hardware devices on the computer. Each "virtual" computer can be started and stopped at will, so long as the exact state of the virtualized hardware (e.g., the processor registers and the contents of memory) are preserved. Although this approach is currently utilized at a very low level, providing "guest" operating systems with entire virtual computers, it is technically feasible and would be potentially interesting to apply this same technique at the application runtime layer. If applications were executed in virtualized environments, it would be possible for the operating system to save and restore each application's entire execution context as needed, enabling activities to be arbitrarily interrupted and resumed. However, this kind of architectural modification would need to be built deep into the operating system itself—far beyond the scope of what would be reasonable for prototyping systems—and it would also introduce additional overhead, since applications used in multiple concurrent activities would need to be managed as multiple, separate instances, consuming valuable system resources.

7.3.2 Applications and Encapsulating Evolving Work

Integration between prototype activity-based tools and fundamental information management tools helps to address a common concern when developing these kinds of systems for deployment and evaluation: leveraging a diversity of existing information resources (e.g., databases of calendar items, to-do lists, and contacts) can potentially reduce the amount of work required by users to populate the system with their own data, thereby reducing their frustration in adopting the system and the time needed to come "up to speed" before more representative day-to-day use of the system can be observed.

Giornata makes extensive use of AppleScript, both as an intra-application communication mechanism—a very common programming practice in the Cocoa and

Carbon APIs—and as a means for taking advantage of applications that are otherwise activity-unaware. The advantages of this application scripting approach are twofold: (1) to maintain the illusion that the activity awareness provided by Giornata is an integrated aspect of the entire operating system and (2) to marshal the assistance of other applications to accomplish more elaborate automation and collaboration tasks than if these capabilities had to be hand-coded into the system.

Not only are most OS X applications easily "scriptable" using AppleScript, but many of the "core" information organization tools that ship with the operating system (e.g., Address Book, iCal) are implemented using a two-tiered architecture. This design separates the tool into a publicly accessible "framework" module that handles data storage and information processing and a high-level wrapper application that serves as the tool's "interface" to the user. As a result, third-party applications can be built to link against the framework and immediately take advantage of all of the tool's information management capabilities, often working seamlessly alongside the user-visible application component.

However, this two-tiered, application–framework architecture is not employed for all critical information management tools. In fact, it is not employed in the case of perhaps the most pervasive information management tool available on the desktop platform: e-mail. While most programming environments feature a rich tool set for accessing stored files, the World Wide Web, and even low-level interface objects like windows, OS X lacks a common architecture, framework, or programming toolkit for accessing or managing e-mail (as, to the best of my knowledge, do all other contemporary operating systems[6]). There are, perhaps, a number of reasons that this is the case: the necessity of supporting a variety of e-mail protocols and handling attachments

[6] One exception is the JavaMail API (http://java.sun.com/products/javamail/, accessed 25 January 2008), although this toolkit is only available when using Java, which, as a relatively high-level language, presents alternative technical challenges for programming tools that integrate deeply with other aspects of the operating system (e.g., interacting with existing applications or providing window management capabilities).

of varying types and encodings are just two obvious examples of technical hurdles that such an approach would present. However, given the ubiquity of e-mail in knowledge work collaboration (see also sections 2.1.3 and 3.3), it is somewhat surprising that steps in this direction have not yet been taken.

Although building stronger links between individual applications and an activity-based infrastructure can provide a more compelling user experience and reduce the "metawork" burden associated with setting up and maintaining activity representations, there still exists a certain degree of friction between the existing desktop paradigm with its strong focus on applications and a true activity-based computing environment. A more radical approach to addressing this struggle for organizational supremacy between applications and activities might be to re-examine the viability of an OpenDoc-style architecture (Apple Computer, Inc., 1995) where applications serve a secondary role as tools that are brought to bear only when needed and in the context of a particular document. This approach would help to transfer some of the focus of contemporary interfaces from the application to individual documents, which could then be organized more readily using activity as a lens. In this configuration, document objects would become responsible for mediating between activity and application, for example, requesting that an application provide a particular set of tools given a certain type of data in the document and the current activity context.

7.3.3 Summary of Technical Requirements for Application Development and Runtime Tools to Better Support Activity-Based Computing

In order to better support the development of activity-based computing systems, future application development and runtime tools should:

- continue to improve existing memory and background process management strategies, increasing the number of applications that can be in use—across multiple activities—at one time;

- require application developers to expose externally-invokable functions that can produce a complete snapshot of the application's runtime state and can resume execution from a previously-stored state upon request;
- provide mechanisms for virtualizing individual application instances, allowing the system to execute each instance within a self-contained runtime "sandbox" that can be paused, suspended to disk, or resumed from a previous execution state as directed by an activity management system;
- encourage application developers to design programs that rigorously follow the model-view-controller paradigm and to provide "hooks" in the controller portion of their programs that allow external applications to script program execution and seamlessly exchange data; and
- include a universally-available scripting language that can be used by activity-based systems to augment existing applications with activity-aware features.

7.4 Discussion and Reflections

The technologies enabling the realization of the Giornata system far surpassed those available for experimenting with previous activity-based computing systems. The resulting prototype required less time to build, integrated much more closely with the operating system and key applications, and was far more stable than its predecessor systems, allowing the system to be deployed and tested *in situ*. The increasing availability of these kinds of technologies in mainstream operating systems is an encouraging sign both for the more widespread development of activity-based tools as well as prototypes that explore other kinds of novel interactions at the desktop level.

However, there are still a number of persistent technical challenges that pose barriers to successfully building and deploying systems that explore these kinds of novel user interface designs. Some of these challenges are actually side effects of those "activity-friendly" technologies described above; in these cases, the availability of

technical characteristics amenable to developing activity-based systems served to emphasize an area in which lack of support reduced the potential benefit. In other cases, the activity-based interaction paradigm itself revealed tensions caused by long-standing assumptions deeply encoded in the operating system and application development toolkits.

CHAPTER 8

CONCLUSIONS

In this chapter, I reflect back upon several aspects of this research, revisiting both the challenges as well as the thesis statement. In addition, I present numerous opportunities for future work in the domain of activity-based computing.

8.1 Revisiting the Challenges

8.1.1 Challenge 1: Activities are a Part of Fluid Work Practice

Providing fluid support for maintaining multiple simultaneous activities continues to be a challenge. Giornata's deployment reiterated the utility of encapsulating work into discrete activities. Many of the study participants cited the virtual desktop feature as being a valuable component of the system, particularly with respect to keeping appropriate information at hand and reducing the overall clutter of both the system's windows and its desktop.

However, the initial assumption that activities are individual and distinguishable turned out to be overly simplistic. Several participants expressed frustration that they could not easily share windows, files, or contacts across multiple activities. In some cases, this was a side effect of the system's underlying application-centric programming model; in those cases where an application manages all artifacts of a single type within one window (e.g., Parallels Desktop, iPhoto, iTunes), it makes sense that users would want to replicate access to this "portal" into their data across multiple activities. In other cases, it was clear that the boundaries between activities were quite permeable, with the same clusters of files and contacts associated with several, closely-knit activities. Since it would seem that a prerequisite for fluidity in the user interface is to have an underlying model that matches the user's mental model as closely as possible, future systems will

need to grapple with creating a more flexible, less discrete model of activity, or, at the very least, formalize some system of relationships among activities so that interrelated activities can more easily share attributes with one another.

Another critical property of activity management tools is their importance in helping to remind the user of ongoing tasks. Giornata, like Apple's Spaces application, provided a convenient overview of all ongoing tasks on demand. However, the lack of easily distinguishable visual characteristics across activities and the requirement that to see the activity overview, an action be taken that occluded the focal activity both imposed artificial barriers for gaining an instant, accurate sense of the state of ongoing activities. While a peripheral-display approach like Kimura seems to be a superior way to provide persistent representations of background activity, this does impose additional systems challenges, either requiring that a distributed architecture be employed to coordinate among multiple machines to generate the focal and peripheral displays or that more robust multiple-monitor support be engineered into a single-machine implementation of such a system. In either case, it is unclear what kinds of visual representations for these background activities would be the most useful; initial studies comparing the various Kimura montage representations proved inconclusive.

8.1.2 Challenge 2: Activities Encapsulate Evolving Work

Giornata responded generally well to the second challenge, eliciting strong affinity among almost all of the participants for per-activity storage and the ability to associate multiple information types with semantically meaningful activities. However, as noted in section 8.1.1, the fact that activities sometimes have "fuzzy" boundaries necessitates consideration of user interface designs that can naturally support activity overlap or the replication of activity contents in multiple activities at the same time.

Supporting evolutionary information classification remains a significant challenge, however. While it was clear that the ability to create a nameless activity and

immediately begin work in that working context without requiring that the user initially specify any of the activity's characteristics was perceived to be a very useful feature of the design, the tagging/retroactive tagging system was utilized much less than originally anticipated. Participants' general lack of enthusiasm for tagging their activities—especially evolutionarily—could be attributed to a number of factors, including a perceived lack of usefulness in the short term (suggesting a need for deploying and evaluating these kinds of systems over very long periods of time) and a belief that search-based navigation is slower and/or more prone to error than browsing-based navigation within a relatively small number of recent activities. Although it is well-acknowledged that this process of filing and categorization can be one of the more difficult aspects of knowledge work (e.g., Lansdale, 1998), tagging might not be the most appropriate mechanism for encouraging incremental or iterative filing. In fact, alternative schemes for organizing artifacts, such as temporally (e.g., Freeman & Fertig, 1995; Rekimoto, 1999) or by contact (e.g., Nardi, Whittaker, Isaacs et al., 2002) might be more effective in the long run.

8.1.3 Challenge 3: Activities are Collaborative

Designing a system that could support both activities and collaboration proved to be one of the most persistent challenges *in the context of the desktop*. Creating a persistent and accessible user interface mechanism for associating contacts with the current activity and accessing relevant electronic communications was not only a significant engineering challenge (due in no small part to the lack of a universal e-mail programming framework), but the resulting user interface was also met with significant resistance from the study participants. The idea of managing contacts and communications on the desktop—a user interface construct typically used exclusively for managing windows and files—seemed to stretch the desktop metaphor uncomfortably far.

Other approaches to activity-based computing have enhanced e-mail clients and web portals, which might be a better fit for supporting activity-based collaborative practices. One of the ongoing challenges for activity-based system designers will be in deciding where in the existing user interface it will be most appropriate to embed activity representations, recognizing that constructing systems that can provide pervasive and consistent activity support throughout every aspect of the user interface will likely require substantial re-examination of the data structures and programming interfaces at much lower levels of the operating system.

The challenges related to preventing information disclosure or resolving differences in the granularity of activity specification were not explored in the deployment of Giornata. It is significant to note that my initial efforts to incorporate these features into the Giornata system were quickly arrested by substantial technical hurdles; I anticipate that these will continue to be significant challenges for the next generation of activity-based computing systems.

8.2 Revisiting the Thesis Statement

This dissertation initially sought to answer the following research questions:

- What are the current mismatches between knowledge work practice and systems support for these practices?

I provided a summary of empirical studies of knowledge workers in section 2.1 and the abbreviated results of a study of knowledge workers' information sharing practices (and breakdowns) in section 3.3. There currently exist a wide variety of mismatches between practices and systems support, which fall into three broad categories: (1) current systems support multitasking and task awareness poorly, laying the burden of managing windows on the user; (2) support for resource organization is generally limited to storage of artifacts in a hierarchical filing system or, increasingly, in an unstructured e-mail inbox; and (3) there is uneven support for collaboration, often

requiring that knowledge workers fall back to the lowest common denominator in order to ensure that the collaboration works as expected.

- How do theories of human activity inform the development and anticipate the potential for innovation in activity-based systems for supporting knowledge work?

In section 2.3, I presented several cognitive and psychological theories of human activity as they relate to the design of activity-based computing systems. I focused on unpacking these theories' stances regarding the relationships among the primary actor, mediating tools, objectives, social structures, plans, and cognitive capabilities as an inspiration for developing appropriate computational models of activity. One of the most useful findings from these theories was the multifaceted representation of activity posited by the activity theorists. This representation based on tools (in the desktop context, applications and information resources), a community of practice (colleagues), and an objective (the semantically-meaningful description of the activity) significantly influenced my activity model in the Giornata system.

- In what ways are existing user interface techniques and metaphors sufficient (or insufficient) for representations of and interactions based around activity?

I presented a survey of activity-based systems and their user interface approaches in section 3.1. Activity-based systems have historically expanded upon two major user interface metaphors: the virtual desktop and the enhanced personal information management tool. However, only recently have desktop systems begun to explore the integration of activity-based resource organization and collaboration tools with the traditional window management capabilities. Many personal information management tools suffer the converse limitations, providing rich support for collaboration and resource organization, but typically offering little support for moment-to-moment multitasking and task awareness.

- How do knowledge workers use activity-based systems in the course of day-to-day work?

In Chapter 6, I presented detailed qualitative and quantitative results from a long-term deployment of the Giornata system. This study represents the only longitudinal study of an activity-based computing system that I am aware of in the research literature, and the results indicated that Giornata was able to support a wide variety of usage patterns, influenced in large part by the variety of participants' existing work practices.

- What are the technical requirements for these kinds of systems?

I reflected upon the enabling technologies that made the realization of activity-based computing systems possible and the persistent technical challenges in creating these systems in Chapter 7. Various features that are now becoming commonplace in modern operating systems and application programming interfaces are making it much easier to develop activity-based prototype systems. Some of these features also provide powerful opportunities for activity-based systems to closely integrate with existing tools and applications. However, some challenges—notably, those with deep connections to the application- and document-based paradigm that is so deeply engrained in contemporary desktop computing systems—continue to pose barriers to the creation of robust, fault-tolerant, activity-based user interfaces.

At the beginning of this dissertation, I proposed the following thesis statement for the study of activity-based computing systems:

> An activity-centric perspective can drive innovation in desktop computing by guiding the development of new user interface capabilities, metaphors, and techniques. These innovations will be better suited for supporting the multitasking, resource organization, and collaboration practices of knowledge workers than existing computational tools.

My adoption of an activity-based perspective inspired the design of three novel prototype systems and suggested combinations of features that were not explored in previous systems also designed to support various aspects of knowledge work. My system designs were grounded in empirical observations of how knowledge workers

manage multiple, simultaneous activities and use these activities to structure their resource organization and collaboration strategies. My designs also incorporated models of activity that were inspired by theoretical perspectives on activity and cognition. My evaluation of the capstone system in the dissertation, Giornata, revealed that participants appreciated the multitasking and resource organization features of the system, expressing both a qualitative preference for these features and demonstrating active adoption and appropriation of the tool in the context of a long-term deployment. However, the system was only reported to be minimally helpful in supporting collaboration practices; while this may be a side-effect of design decisions and a few minor user interface bugs, the relationship between activity and collaboration will require more in-depth investigation.

8.3 Future Work in Activity-Based Computing Systems

This research suggests a number of potentially fruitful research agendas moving forward, each investigating different aspects of activity-based system development or adoption. They include: systems in which activity is integrated deeply into the underlying desktop metaphor, the use of activity representations to bridge individual work on the desktop and collaborative work on the Web or through a personal information management portal, further user studies to better understand perceived activity lifecycles and activity tagging behaviors in different settings, and the application of activity representations as a unifying construct in ubiquitous computing environments.

8.3.1 Activity Permeating the Desktop

One of the features most valued by the user study participants was Giornata's ability to seamlessly connect the construct of semantically meaningful, user-specified activities with the very common action of storing documents on the desktop. The system's design resulted in much tighter integration between the desktop and the activity

representations stored in the system than was available in previous activity-based systems.

What Giornata did not accomplish as successfully was providing a user interface incorporating representations of colleagues on the desktop. Study participants were critical of the Contact Palette interface as being too heavyweight for regular use. Additionally, because the action of dragging colleagues' icons from the Address Book into a special-purpose palette did not build on an existing interaction technique, participants did not have an existing motivation for making contact–activity associations. One potentially fruitful research agenda would be to integrate the system's activity representations into other tools that are already central to collaboration tasks (e.g., personal information management tools or e-mail clients), enhancing these tools with activity representations while building on existing interaction practices.

From an interaction design perspective, Giornata's activity representations were useful for *visually simplifying* the desktop. When activity representations were used to determine the visibility of system windows—an approach common to most virtual desktop systems—the result was a reduction in visual clutter on the desktop since only those windows relevant to the current activity were visible and able to compete for the user's attention. In the Giornata system, this principle was also applied to the contents of the desktop; Giornata also reduced clutter on the desktop because only those items relevant to the current task were displayed on the desktop surface. This observation suggests that one opportunity for integrating representations of activity into personal information management applications would be to apply a similar design principle of simplification. For example, what might an e-mail client look like if were to only display e-mails from colleagues that you had associated with the current activity, provided user interface shortcuts for sending information to these "relevant" colleagues, and only provided incoming e-mail notifications in the Dock when new messages arrived from these individuals? Giornata provided some of these features by using AppleScript to

control the Apple Mail application, but in order to take full advantage of this type of activity-centric interface simplification, representations of activity would need to be incorporated much more deeply into collaborative applications.

8.3.2 Activity as a Bridge Between Individual and Collaborative Work

Prior research in activity-based computing systems can essentially be divided into two categories: those that embed representations of activity on the desktop (e.g., Rooms, Task Gallery, Kimura, GroupBar, Scalable Fabric, TaskTracer, WindowScape, Giornata) and those that embed representations of activity in a personal information management tool or e-mail client (e.g., Haystack, TaskView, ContactMap, Taskmaster, UMEA, ActivityExplorer, UAM). There are legitimate justifications for placing the focus of an activity-based system in one or the other of these categories. If the system is designed to focus on providing activity support primarily on the desktop, the system will be able to monitor user interactions across all running applications, manage windows and local files on a per-activity basis, and—in all likelihood—store all of the potentially sensitive window interaction history data on the local computer. If the system is designed to focus on providing activity support from within a personal information management tool, then the system is in a much stronger position to share activity information with collaborators (potentially enabling the creation of an organization-wide, activity-based knowledge management repository), to manage files, contacts, and communications on a per-activity basis, and offer the option of storing activity information locally or on a shared collaboration server (e.g., Lotus Domino or Microsoft Exchange).

Both of these approaches have their advantages, but, currently, there exists no system that uses activity representations to serve as a "bridge" between these two very different implementations. A hybrid system could, for example, provide support for managing activities at the desktop level and continuously communicate the local activity state information to the collaborative server, ensuring that interactions with the personal

information management tool or e-mail client reflect the same knowledge of the user's activity. This "activity bridge" approach could work in the inverse direction as well, allowing collaboration tools to "push" state information from the server to a local activity manager, which could then suggest a change to a relevant activity if a critical e-mail were to arrive or a calendar appointment were about to become active.

Although this hybrid approach would potentially provide advantages unavailable in current activity-based computing systems, designing such a system would require extensive engineering research into the selection of communications protocols and the development of an activity synchronization infrastructure to ensure that all system components are kept apprised of the current activity state.

8.3.3 Understanding Activity Lifecycles

One of the most interesting findings from the Giornata user study was the fact that several participants characterized their activities as existing in one of many nuanced states, including "open," "closed," "dormant," and "background." This stands in contrast to the comparatively naïve, binary model that Giornata used internally: activities are either "open," and therefore accessible, or else they are permanently "closed." This difference in the perception of an activity's lifecycle is significant, as the more complex suite of options proposed by the study participants suggests that they have a much more acute awareness about how and when they anticipate resuming those activities in the future.

This criticism of an overly simplistic activity model has been leveled before. In section 2.3.1, I discussed a series of extensions to Activity Theory proposed by Boer, van Baalen & Kumar (2002), which were intended to better describe the lifecycle of activities than the largely static representations previously used in the community. The proposed changes focused on acknowledging the relationships among activities—both those

occurring simultaneously and those undertaken one after the other—in order to better explain how activities ebb and flow over time and exert influence upon one another.

This concept of activity lifecycles would be a good candidate for further empirical study. Some of the research questions that need to be answered in the interest of developing more appropriate activity representations in future systems include: *What happens to activities over time? When do users decide to begin a new activity? Are new activities really new (following a Giornata-like "clean slate" model), or do they typically spin off from other, previous activities (following a UAM-like template model)? Do users conceive of their activities as being hierarchical? When are activities truly "finished"…if ever?*

8.3.4 Investigating the Semantics of Tagging

Activity tagging was one of the less-utilized features of the Giornata system, for a variety of reasons. However, I did see some consistency in the way that participants were applying tags to their activities: tags were primarily descriptive, activities' tags were typically unique and orthogonal to all other activities' tags, and, excluding project names, there appeared to be little intentional use of memorable, "keyword"-type tags that would be useful in finding components of an activity using the operating system's Spotlight search capability. Anecdotally, these tagging behaviors seem to run quite counter to the ways that tags are used in primarily collaboration-oriented environments (e.g. Flickr). This raises an important question: do knowledge workers conceive of tags as serving fundamentally different purposes when they are tagging artifacts for their own reference as opposed to tagging public or shared artifacts? In the current instantiation of Giornata, this is not a critical issue, as it is somewhat rare to share an artifact with another user in such a way that Giornata's tags are preserved. However, in future versions of the system, it is easily conceivable that multiple Giornata users might want to collaborate together, in which case their tagging behavior might have to serve two simultaneous purposes:

functioning as an personal reminder of the significance and meaning of the activity and as a means of communicating meaning and intent to the remote collaborator.

This open question provides another opportunity to conduct an empirical enquiry for better understanding users' tagging behaviors in a variety of contexts. In contrast to the previous future work proposal, studying users' perceptions of the activity lifecycle, this question might lend itself more naturally to a controlled laboratory study or a focus group, since it would be very useful to be able to draw generalizations across multiple participants' responses when exposed to different tagging scenarios.

8.3.5 Ubiquitous Activities

Mobile devices originated as single-function devices (e.g., mobile telephones); these devices were generally designed to do one task well. As these devices became more sophisticated and began to integrate an increasing number of features, they grew to function more and more like general-purpose computers in a small form factor. However, even though these devices' capabilities continue to expand, the user interface available for interacting with them continues to be constrained by the size of the device and the types of inputs it can process.

Because activities provide a natural scoping mechanism for the resources and contacts relevant in a given context, representations of activity are likely to be useful when employed on devices with limited display and interaction capabilities. Just as activity-based computing can serve to drastically simplify the appearance of a cluttered desktop otherwise full of windows and information resources, a research agenda focused on adapting an activity-based infrastructure for mobile devices might help to constructively constrain the applications and input options available on these small devices at any given time, based on the user's current "activity" or context of use. This type of infrastructure might also provide opportunities for establishing context-sensitive

connections between mobile devices and other activity-aware computing systems, enabling the development of a new class of novel interaction techniques.

APPENDIX A

GIORNATA USER'S GUIDE

Giornata User's Guide

Giornata version 0.5
17 July 2007

Stephen Voida
svoida@cc.gatech.edu

NOTE: The contents of this document are subject to change as new features are developed and released in subsequent versions of the Giornata software. Please make sure that the version of the user's guide that you are using as a reference matches the version of the software running on your computer.

Figure A.1 Page one of the Giornata User's Guide,
as provided to study participants: Cover page.

Contents

Figure A.2 Page three of the Giornata User's Guide, as provided to study participants: Table of contents.

About Giornata

Giornata is a program for Mac OS X that provides a different way of storing, organizing, and retrieving your information.

In most computer systems, you have a single "desktop" that holds all of your open windows. You are responsible for opening, closing, and finding windows when you need them. It's also up to you to decide where to store the files you create as you work. In addition, in most computer systems, the only place where the people you are working with appear are in your email inbox and in your address book.

Giornata uses a different metaphor for using your computer. In Giornata, you organize the things that you do into *activities*, such as writing a letter, editing your web pages, or balancing a budget. Each of these activities has its own desktop, where only the windows that have to do with completing the activity are displayed. In Giornata, in addition to keeping your files organized in folders, you can also store the files that have to do with each activity right on the desktop (since you have different desktops for different activities, doing this won't clutter up your desktop as much as in traditional computer systems). Finally, if you are working with others to get your activity done, you can link these people to the activity and have access to tools for keeping in contact and sharing information with them.

The easiest way to understand how working with Giornata is different is by seeing the program in action. You can watch a short video (about 5 minutes long) that illustrates the features of the Giornata system by visiting the following web site with your computer:

http://www.cc.gatech.edu/~svoida/pubs/giornata-uist07.mov

(This video was produced for scientists with an interest in what makes the Giornata system work, so some parts might provide more of a technical focus than you need. Nevertheless, it does provide a good overview of how the system looks and works.)

This guide provides an overview of how to install, use, and troubleshoot many of the problems you might encounter with the Giornata system. If you have problems or questions that this guide does not address or if you have suggestions about how the Giornata system might be improved, you can contact the researchers by email at **svoida@cc.gatech.edu** or by mobile phone at [redacted] at any time.

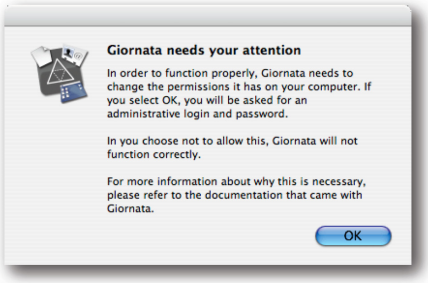
Figure A.3 Page five of the Giornata User's Guide, as provided to study participants: Description of the system.

Using Giornata

Starting Giornata

Start the application by double-clicking the Giornata icon in your Applications folder:

The first time you run Giornata or after you update the program using the “Check for updates” tool, you may see the following dialog appear once or twice shortly after you start the program:

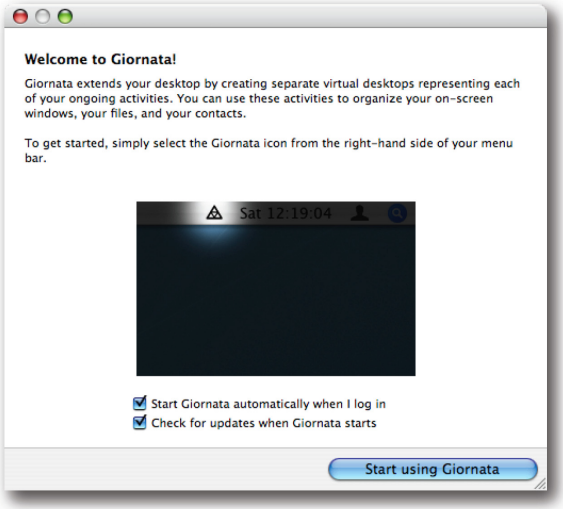


There are two parts of the Giornata program that require administrative permissions to run correctly: the part of the program that attaches to your Dock so that Giornata can hide and show windows and the part of the program that monitors your home folder for changes (for example, when you save new files to the desktop). After you click “OK”, Giornata will ask for your password (or an administrator’s password, if your account is a Standard user account).

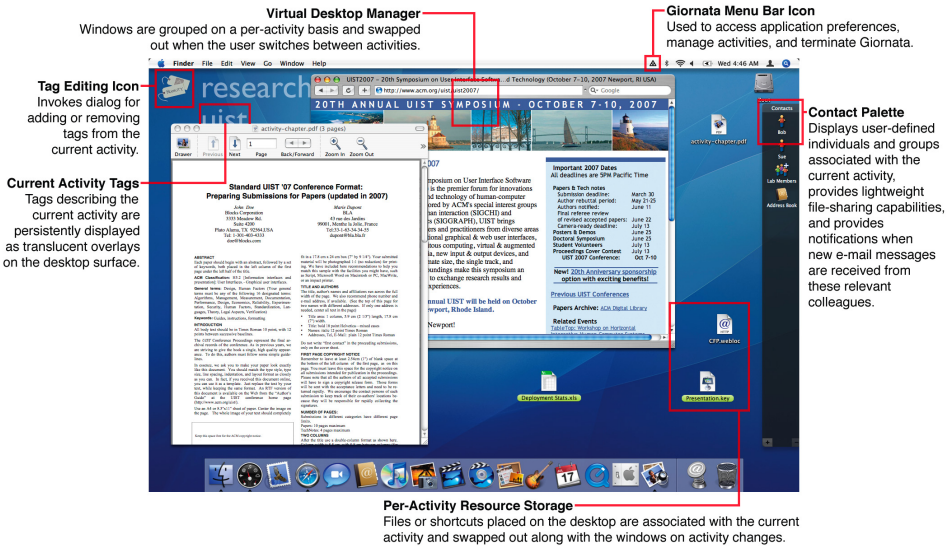
The program may also ask to move any items on your desktop to another folder in your home directory. In Giornata, you use the desktop as the place to store files associated with each of your activities, so moving any items out of the way when the program starts prevents Giornata from incorrectly associating these items with an activity once the program starts running.

Finally, the first time you run Giornata, you should see a welcome dialog box with some information about the program and a few options that you can set:

Figure A.4 Page six of the Giornata User's Guide, as provided to study participants: Instructions for use.



Once Giornata is running, you will have access to a number of features for managing your activities and the information associated with them. Here is a screenshot Giornata's features:

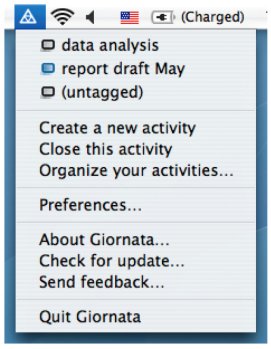


In certain instances, Giornata will behave strangely after the first start-up until you quit the program and re-start it again. (This is a bug that we're still trying to fix.) If Giornata doesn't work as described here during the first run, please quit and then restart the program to fix the problem.

Figure A.5 Page seven of the Giornata User's Guide, as provided to study participants: Instructions for use.

Stopping Giornata

Most of Giornata's functions can be accessed using keyboard shortcuts or choosing items on the status bar menu (⚠) on the right-hand side of your menu bar. To stop Giornata and preserve the current state of your activities, choose the "Quit Giornata" menu item on this menu:



When you quit the program, all of the items on the desktop are stored in their corresponding activity folder on the hard disk and all windows (in all activities) are brought to the front, so that you can continue to access them while Giornata is not running. When you restart the program, Giornata will attempt to place these windows back on the correct desktops and resume the activity in which you were last working. If you have opened new windows between stopping Giornata and restarting it again, these windows will remain visible in whichever activity is resumed until you manually close them or move them to the appropriate activity.

Creating a New Activity

To create a new activity, you can either choose the "Create a new activity" menu item on the Giornata status bar menu, or you can use the keyboard shortcut: ⌘⌥= (command+option +equals). All Giornata keyboard shortcuts include both the command and option keys; you can remember this shortcut because it is the combination of these two keys plus the key with the "plus" sign on it (for "add" a new activity).

When you create a new activity, you will see a clean desktop without any windows or files. (You will still be able to access your hard disk and any mounted volumes, as usual). New activities will also not have any tags associated with them; the desktop will display the message "(untagged)" in the tag display area.

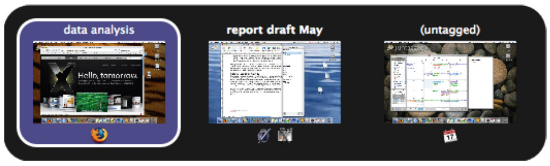
Switching Between Activities

There are a number of ways to switch from activity to activity:

- You can view a list of open activities by clicking on the Giornata status bar icon (⚠). These activities will appear at the top of the menu that appears; selecting one of these items switches to the corresponding activity.

Figure A.6 Page eight of the Giornata User's Guide, as provided to study participants: Instructions for use.

- You can pop up a quick activity switcher, which displays a thumbnail view of all open activities, by pressing ⌥⇥ (option+tab). The quick activity switcher works similarly to the regular application switcher built into OS X and looks like this:



  The quick activity switcher remains on screen until you release the option key. While the quick activity switcher is displayed, you can type ⇥ (tab) or → (right arrow) to select the next activity to the right or ⇧ ⇥ (shift+tab) or ← (left arrow) to select the next activity to the left, click on a thumbnail to switch to the corresponding activity, or drag and drop the thumbnails with the mouse to re-order to activities however you desire.

- You can use the keyboard shortcuts ⌘⌥← (command+option+left arrow) and ⌘⌥→ (command+option+right arrow) to quickly switch to an activity immediately to the "left" or "right" of your current activity. (The left-to-right ordering for the activities is as displayed in the quick activity switcher. Note that the activity list "wraps around;" that is, moving "right" from the rightmost activity switches to the leftmost activity.)

When you switch between activities, all of the windows, files on the desktop, contacts in the Contact Palette, and tags are updated to reflect the status of the new activity. A visual transition effect is also (optionally) displayed, and can be configured using the "Preferences…" menu option on the Giornata status bar menu.

Adding Tags to an Activity

Tags are used to add any number of words to an activity to help describe what that activity is about. These tags allow you to identify an activity among the others you are working on, and they are used to add some additional meaning to the files that you "touch" while working on the activity: once an activity is tagged, all the files you change while working in the activity are "tagged" with the activity's tags. These tags can be used as search terms with Spotlight—if you use a tag as a search word, all files with that tag are included in the search results listing.

You can add (or edit or remove) tags from an activity using the tag editor displayed on the desktop background (beneath all of the open windows). An easy way to reveal the tag editor is to use Exposé to reveal the desktop: press F11 to temporarily move all of the windows out of the way.

Click on the manila tag icon ( ) to reveal the tag editing dialog box:

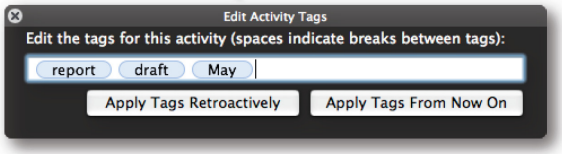


Figure A.7 Page nine of the Giornata User's Guide, as provided to study participants: Instructions for use.

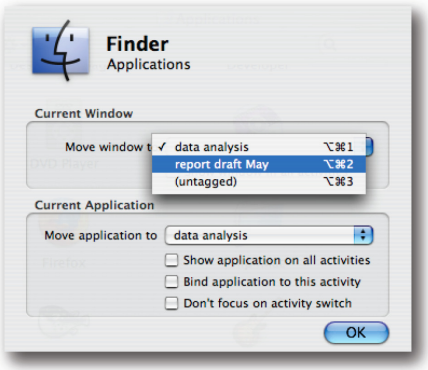
This box will be populated with a number of “token”-style items, one for each tag currently set describing your activity. You can manipulate these tags just like email addresses in a new email message in the OS X Mail application.

There are three options for exiting the tag editor dialog. You can click the small ‘X’ icon in the upper left corner of the box to dismiss the editor without making any changes to the activity’s tags. You can also click either of the two silver buttons to accept your changes and dismiss the editor dialog.

If you choose the “Apply Tags From Now On” option, the tags will change on the desktop and all files touched from that point forward will inherit the new set of tags; all previously-touched files’ tags will stay the same. If you choose the “Apply Tags Retroactively” option, the tags will change on the desktop and all files touched from that activity, past and future, will inherit the new set of tags. The second option allows you to add tags to your work a little at a time, while ensuring that all files reflect the most recent “description” of your work.

Moving Windows Between Activities

You can easily move a window from activity to activity, or force it to be displayed in all activities (sometimes useful for applications like Mail). Select the window to bring it to the front, and then press ⌘⌥o (command+option+letter ‘o’) to display the options sheet for the window:



You can move the one, current window to another activity by choosing the destination activity in the top drop-down menu (the move will take place as soon as you make a selection). You can also move all windows of the current application to another activity (or to all activities) by choosing the destination activity or clicking the box next to “Show application on all activities” in the bottom section of the options sheet.

Associating Files with an Activity

To associate a file with an activity, simply save (or move) it to the desktop while working in the activity. All files on the desktop are automatically saved to a folder associated with the activity when you switch to a new activity or quit Giornata.

If you wish to continue storing files in another location on your hard drive (e.g., to simplify accessing the file from multiple activities), go right ahead. All files that you touch while working

Figure A.8 Page ten of the Giornata User's Guide, as provided to study participants: Instructions for use.

on an activity are tagged with the activity's tags (so long as they are stored *someplace* within your home folder). This allows a single file to be shared among multiple activities, in which case it will be annotated with all of the tags from all of the activities sharing it—without you having to do any additional work or filing.

Associating Contacts with an Activity

In addition to windows and files, you can also associate contacts from your Address Book with an activity. When you associate a contact with your activity, you enable four collaboration features in Giornata:

1. An at-a-glance visualization reflecting the number of unread emails you have in your email inbox from each of your associated contacts;
2. Two-click access to the contact's Address Book card;
3. Two-click access to a filtered view of your inbox, showing only the messages from the selected contact; and
4. Drag-and-drop file sharing with individual contacts or groups of contacts.

Just like windows and files, these contacts are automatically swapped out when you switch between activities, so that you always have the contacts relevant to a particular activity immediately at hand.

In the Contact Palette, icons represent individual contacts or groups of contacts. By default, one group is available: Address Book, which contains all of the people and groups in your OS X Address Book. (No information about the contents of your Address Book will be shared with the researchers.)

The icons in the Contact Palette are decorated with small "badges" that provide at-a-glance awareness about the contents of the palette. Groups with at least one "member" have a small, white plus in the upper right corner. When you have unread emails in your email inbox from an individual contact, their icon is badged with a number in a red circle indicating the number of unread emails they have sent. Groups display the number of all unread emails you have received that were sent by members of the group.

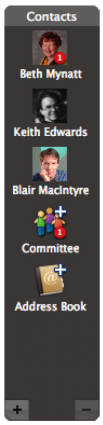
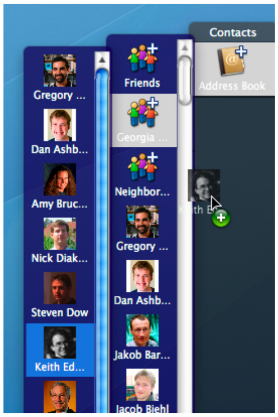


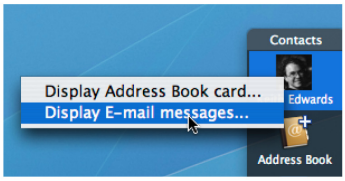
Figure A.9 Page eleven of the Giornata User's Guide, as provided to study participants: Instructions for use.

You can expand a group by clicking once on its icon; a pop-up palette appears to the side of the original palette, displaying the members of the group. If you move your mouse away from the pop-up menu or click on another item in the original palette, the pop-up palette automatically disappears.

To associate a contact with an activity, you can drag individual contacts or groups of contacts from your Address Book group (in the Contact Palette) to the empty space in the Contact Palette:



To quickly access a contact's Address Book card, click once on the contact to bring up a context menu and select the "Display Address Book card…" item. To see a filtered view of your inbox containing only the emails sent by a contact, choose the "Display E-mail messages…" item instead:



To create a new group, click on the "+" button in the bottom left corner of the Contact Palette and a new, empty group will appear. Click on this group's icon to edit the group's name. You can add members to groups by dragging and dropping contact icons onto the group directly.

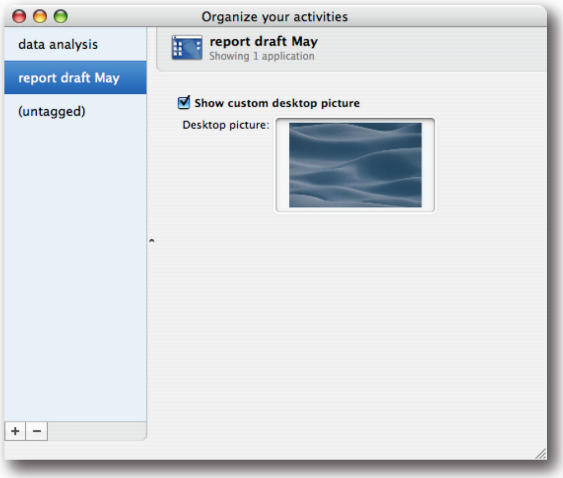
To remove an icon from the Contact Palette or from a group you created, click twice on the icon (once to select it and a second time to dismiss the pop-up menu item), and then click the "-" button in the bottom right corner of the palette.

Figure A.10 Page twelve of the Giornata User's Guide, as provided to study participants: Instructions for use.

Changing an Activity's Wallpaper

An easy way to improve the visual distinctness of your activities (especially when using the quick activity switcher or paging through your activities quickly) is to apply a custom wallpaper to one or more of them.

By default, each activity is created using whatever desktop wallpaper you were using when you started the Giornata program. You can customize the wallpaper for any activity by choosing the "Organize your activities…" menu item from the Giornata status bar menu (⚠). The following dialog box will appear:



Select the activity you wish to customize in the source list on the left, turn the "Show custom desktop picture" check box on, and then drag and drop a picture from anywhere on your computer onto the miniature desktop image in the dialog box to set a new wallpaper picture. Any changes you make in this dialog will take effect immediately.

You can remove a custom activity wallpaper by opening the "Organize your activities…" dialog box and turning the "Show custom dialog picture" check box off.

Closing an Activity

When you have completed an activity and wish to remove it from the list of activities in progress (including the quick activity switcher and the Giornata status bar menu), you can close the activity. **Closing an activity only removes the activity from the list of current activities in Giornata; all of the files on that activity's desktop are *saved* on the computer and remain accessible through the Finder and from Spotlight searches using the activity's tags.**

You can close an activity by switching to the activity and choosing the "Close this activity" menu item from the Giornata status bar menu (⚠). You will be immediately switched to the activity on the left and the closed activity will no longer show up on the lists of open activities in Giornata.

Figure A.11 Page thirteen of the Giornata User's Guide, as provided to study participants: Instructions for use.

Note: Currently, Giornata does not have the capability to automatically close the windows in the closed activity. For the time being, these windows simply follow you to the activity on the left when the switch takes place. This will be fixed in a forthcoming version of the software.

It is not possible to "undo" closing an activity, and the contact list associated with the closed activity will be permanently lost. However, the folder corresponding to the closed activity's desktop will still be available inside the "Activities" folder in your home directory:

The tags will also remain on any files that you touched while working in the closed activity. As a result, you can still use these tags to find the files associated with the closed activity using Spotlight.

Sharing Files with Your Contacts

You can drop files from the desktop directly onto a contact (or group) in the Contact Palette to share that file with the intended recipients via email. When you drop one or more items onto the palette, an email composition window appears, pre-addressed to the appropriate recipients and with the file(s) already attached—all you have to do is edit the message to provide any additional context and click the "Send" button.

Managing Giornata's Preferences

The Giornata software has a number of configurable options that can be controlled using a preferences interface similar to that used in other Mac OS X applications. To access Giornata's preferences dialog, select the "Preferences…" menu item on the Giornata status bar menu.

Giornata's preferences fall into four broad groups, which are selected using the source list on the left side of the dialog box:

- *Application*, including overall options for the software, such as startup, notification, and automatic update preferences;
- *Appearance*, which controls the visual transitions used when switching between activities;
- *Triggers*, which allows customization of the keyboard shortcuts used in the program; and
- *Contact Palette*, where appearance and behavior options specific to the Contact Palette are located.

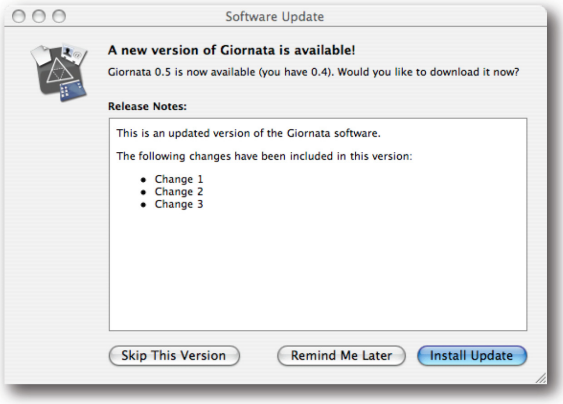
Changes to all preferences except "Contact Palette Display Location" take effect immediately. Due to the complexity of the programming used to hide and show the Contact Palette, you must restart Giornata to see the effect of changes to the palette's display location.

Keeping Giornata Up-to-Date

Giornata includes a mechanism for updating itself when new versions of the software are made available. You can manually check for new updates by selecting the "Check for update…" item on the Giornata status bar menu. When a new update is available, a dialog similar to the following will appear, listing the things that have been changed in the new version of the software and providing options to download and install the update immediately

Figure A.12 Page fourteen of the Giornata User's Guide, as provided to study participants: Instructions for use.

(recommended), to remind you about the update later, or to skip the update entirely (not recommended):



Uninstalling Giornata

Giornata is a relatively self-contained application and can be easily removed from your computer at the conclusion of the study. To remove Giornata from your system, follow these five steps:

1. Remove the following files from your computer by dragging them to the Trash:
   • /Applications/Giornata.app
   • /Users/*(your username)*/Library/Caches/Giornata
   • /Users/*(your username)*/Library/Logs/Giornata.log
   • /Users/*(your username)*/Library/Logs/CrashReporter/Giornata.crash.log
   • /Users/*(your username)*/Library/Preferences/edu.gatech.inspace.Giornata.plist
2. **DO NOT** drag the "Activities" and "Previous Desktop Contents" folders in your home directory (if present) to the trash without moving all of their contents elsewhere on your hard disk. These folders contain all of the files associated with your activities and are not backed up anywhere else on your computer.
3. If you have a "Previous Desktop Contents" folder in your home folder, copy its contents to your Desktop. Once the "Previous Desktop Contents" folder is empty, you may drag it to the Trash.
4. (Optional) Go into the "Activities" folder in your home folder and empty out each activity folder inside of it, either by moving the entire activity folder to a new location on your disk or by moving its contents and then deleting the remaining empty folder. Only when you have relocated (or deleted) the contents of each activity folder inside can you safely delete the "Activities" folder by dragging it to the Trash.
5. Empty the Trash.

Figure A.13 Page fifteen of the Giornata User's Guide, as provided to study participants: Instructions for use.

Frequently-Asked Questions

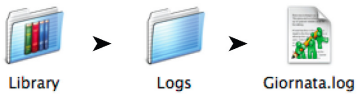
"Giornata"*? That's an unusual name. What does it mean?*
Giornata is Italian for "day's work." This phrase is used to denote the time during the day that work takes place—the time that we expect that a system like Giornata would be useful. In the context of *buon fresco* (wet plaster) painting, a style that was popular during the Italian Renaissance, it also denoted the physical region of a painting that could be completed in a single session before that area of plaster dried. We also liked the image that it invoked combining physical space and activity, something we hope to continue exploring as we make further refinements to Giornata's user interface.

Why is the Giornata status bar menu a triangle?
The Giornata system is based, in part, on a theory of human cognition called *activity theory,* developed by Russian developmental psychologists Lev Vygotsky and Alexsei Leontiev in the early 20th Century. The icon used for the Giornata status bar menu (▲) is a very small version of a well-known illustration of the relationships captured by activity theory, drawn by Yrjö Engeström.

What information does Giornata collect about me? Where do I find this information?
Giornata only collects information about the ways that you use the Giornata software. More specifically, it notes the date, time, and any relevant activity tags when:

- you start the Giornata application
- you quit the Giornata application
- you create a new activity
- you change the tags that describe an activity
- you switch between activities
- Giornata experiences an error condition or crashes

All of this information is stored in a file named *Giornata.log*, which is kept in your home folder. You can see (and delete) any information collected about you if you open your home folder, followed by these icons:



By default, the *Giornata.log* file will open using a program called *Console*. This program is good for seeing the information collected about you, but bad for editing any information that you would rather not disclose. (Your only option in *Console* is to clear all contents of the log file.) You can also drag the *Giornata.log* file to your word processor icon in the Dock and edit it there yourself.

Can I use Giornata with multiple monitors?
The current version of Giornata should work fine on systems with multiple monitors. However, the entire desktop space (across all monitors) is associated with one activity at a time. It is not currently possible to divide the display space into multiple activities or to display multiple activity desktops simultaneously.

Figure A.14 Page sixteen of the Giornata User's Guide, as provided to study participants: Frequently asked questions.

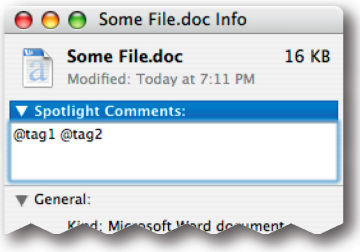
Troubleshooting Problems

I forgot to change my activity and started working on something new. How do I move files to another activity?
The simplest way to move files between activities is to move them from your desktop to a temporary folder, switch activities, and then move them back to the desktop again. *Note: Moving files to the desktop in the correct activity will add the new activity's tags to the files, but if you wish to remove the old activity's tags, follow the directions in the answer to the next question.*

Searching for activity tags with Spotlight is bringing up files that don't have anything to do with that activity. What can I do to fix this?
By default, Spotlight searches for activity tags on files *in addition to* other information stored in the file, such as its name, contents, and other searchable metadata. To limit the search results to activity tags, you can use a special format for your search: placing each tag in double quotes, and preceding each tag name with the at character (@). For example, to search for all files tagged with *tag1* and *tag2*, you would run the following search in Spotlight:

It is possible that a file you find with Spotlight may have different tags on it than you might expect (e.g., if you change what activity a file belongs in, as in the previous troubleshooting question). When you remove tags from an activity, even when you choose to retroactively change the tags, the tags are not removed from the files (in case you need to find something but only remember the tags that were in use when the file was created or edited). Additionally, it is possible (although unlikely) that Giornata might mis-tag a particular file.

You can manually edit the tags applied to any file by selecting that file in the Finder, choosing the File | Get Info… menu command (or pressing ⌘I), and then editing the contents of the Spotlight Comments box displayed for the file:



Note: Giornata stores tags in the Spotlight Comments field using the format *@tag @tag @tag,* with a single space between each tag.

I can't find a file that used to be on my desktop. What do I do?
If the file was on your desktop before you began using Giornata, it was probably moved out of the way to prevent it from being incorrectly associated with one of your activities. Check the folder named "Previous Desktop Contents" in your home folder for the file.

Figure A.15 Page seventeen of the Giornata User's Guide, as provided to study participants: Directions for troubleshooting.

If you can't find the file there, cycle through all of your open activities (you can press ⌘⌥→ repeatedly to visit each of your activities in turn) and check each activity desktop for the file. (Exposé's "Desktop" capability comes in handy here; press F11 to move all of the open windows out of the way for a moment.) If you still don't see the file you're looking for, try looking in the Activities folder in your home directory:

Each activity you have ever created has a corresponding folder inside (named using the activity tags, when possible), and these folders hold the contents of that activity's desktop when it is inactive. Your file will be in one of these locations.

Giornata crashed and now I can't find a window that I had open. What do I do?
When Giornata crashes, the windows that you had open in background activities cannot be accessed until you restart Giornata. Just open the program again, and you should be able to switch activities and find your missing windows.

If Giornata is crashing before you can switch to the activity associated with your window, you can also quit and re-open the application to make your windows visible again. *Note that you may lose unsaved data if you exercise this option.*

Giornata crashed and now it's bugging me about files on my desktop when I try to restart it. What do I do?
Because Giornata uses your desktop as a place to store files specific to the activity you are engaged in, the program will ask if it's okay for it to move any files that are on your desktop out of the way when it starts:

These files will be moved into a special folder in your home folder, named "Previous Desktop Contents":

When Giornata crashes unexpectedly, it doesn't have a chance to move the contents of your desktop to the correct Activity folder for safe keeping and it may leave stray files on your desktop. In this case, the best way to get started again is to allow Giornata to move the files

Figure A.16 Page eighteen of the Giornata User's Guide, as provided to study participants: Directions for troubleshooting.

out of the way (click the “Move Desktop Contents” button when prompted), switch to the activity you were working on before Giornata crashed, and then move the files from the “Previous Desktop Contents” folder back onto your desktop.

I’m still having trouble with Giornata…or I have an idea about what might make Giornata better. What do I do?

We understand that you are taking a risk by participating in our study and using our software. We know that Giornata is certainly not free of bugs and that it has not been tested as thoroughly as a commercial software package. We also know that your first priority when using your computer is in getting your work done, not in having to work around technical problems caused by our software or struggling to make sense of some aspect of our design that’s constantly working *against* you.

During the course of the study, please do not hesitate to contact the researchers at any time of the night or day with problems, questions, or suggestions. The researchers will be available to help you throughout the entire study by email at **svoida@cc.gatech.edu** or by mobile phone at [redacted]. We are committed to making sure that your agreement to participate in this study does not compromise your ability to get things done, and we will do everything possible to ensure that this is the case.

Figure A.17 Page nineteen of the Giornata User's Guide, as provided to study participants: Directions for troubleshooting.

Giornata Keyboard Shortcuts

⌘⌥←	(command+option+left arrow)	Switch to the activity on the left (according to the order displayed in the quick activity switcher)
⌘⌥→	(command+option+right arrow)	Switch to the activity on the right
⌘⌥=	(command+option+equals)	Create a new (empty) activity
⌘⌥⌫	(command+option+delete)	Close the current activity
⌘⌥o	(command+option+letter 'o')	Display an options sheet ('o' stands for "options") for managing the current window (for example, to move it to another activity)
⌥⇥	(option+tab)	Display the quick activity switcher (this stays open until you release the option key or until you press space or enter to confirm your activity selection)

Inside the quick activity switcher (while holding down the option key):

⇥	(tab)	Select the next activity to the right
→	(right arrow)	Select the next activity to the right
⇧⇥	(shift+tab)	Select the next activity to the left
←	(left arrow)	Select the next activity to the left

Researcher Contact Information

Stephen Voida, Ph.D. candidate
GVU Center, College of Computing
Georgia Institute of Technology
Technology Square Research Building
85 5th Street NW, Room 332
Atlanta, GA 30332-0760 USA

Mobile: [redacted]
Office: (404) 894-4488
Email: svoida@cc.gatech.edu
AOL Instant Messenger: [redacted]
Google Chat: [redacted]

Figure A.18 Page twenty of the Giornata User's Guide, as provided to study participants: Keyboard shortcut reference and researcher contact information.

APPENDIX B

GIORNATA USER STUDY SEMI-STRUCTURED INTERVIEW PROTOCOLS

B.1 Midpoint Semi-Structured Interview Protocol

1. What activities have you been working on since you began using Giornata? Are you continuing to work on these activities? What are their current states?
2. Show me how you've been using Giornata for the last *X* days.
3. Have you created any new activities? Have you closed any? Do you move between activities, or do you primarily work out of one or two? If so, which?
4. Did you use…

   …the tagging capabilities of Giornata? For what?

   …the Contact Palette? For what?

   …the quick activity switcher? For what?

   …other features of the system?
5. Have you noticed doing anything differently since Giornata has been installed on your computer? The way you manage your documents? Your communications? The rhythm of your work?
6. Are there aspects of Giornata that have caused frustrations with the way that you work? Are there ways that your thought Giornata could be improved or changed to better fit the way you work?
7. About how much of the time have you been working with Giornata turned on? Did you have to shut it down for any reason? Did it crash?
8. In general, how do you feel about organizing your artifacts around your activities?
9. Anything else?

B.2 Summative Semi-structured Interview Protocol

1. Revisit activities described in the initial interview. Are you continuing to work on these activities? What are their current states?

2. Have you continued using Giornata after the previous interview? If so, how have you continued to use the software? If not, why did you stop? Have you missed having access to Giornata's features? Which ones?
3. For a specific activity that you created during your use of the Giornata system, what did you store on the desktop associated with the activity? What information did you store elsewhere on your computer? Was there a particular criterion that you used to determine where you would store the information associated with the activity?
4. When you collaborate with others, where do you store the files that you use over the course of the collaboration? On your computer? On a website? In an e-mail client? In your filesystem?
5. If you were looking for some content that you knew existed on your computer and was related to activity *X*, how would you find it?
6. For a specific activity that you created during your use of the Giornata system, what content did you share with others? With whom did you share the information? Did you associate these individuals with the activity using the Contact Palette? What mechanisms did you use to share that content? Can you imagine ways that Giornata could have made that sharing easier or more natural?
7. On a scale of 1 to 5, where 5 represents "very useful" and 1 represents "not at all useful," how would you characterize the version of Giornata that you used? Can you elaborate?
8. On a scale of 1 to 5, where 5 indicates that the system allowed you to organize and manage your activities very fluidly within the work that your do from day to day and 1 indicates that the system interrupted the way that you work with its tools, how would you characterize the version of Giornata that you used? Can you elaborate?

9. On a scale of 1 to 5, where 5 indicates that the system helps you to organize your computing information in ways that match very well with the ways that you mentally organize your work and 1 indicates that the system does not help you to organize your computing information in ways that match with the ways that you mentally organize your work, how would you characterize the version of Giornata that you used? Can you elaborate?
10. On a scale of 1 to 5, where 5 indicates that the system helped you to collaborate with your colleagues effectively within your activities and 1 indicates that the system made it more difficult to collaborate with your colleagues, how would you characterize the version of Giornata that you used? Can you elaborate?
11. Anything else?

REFERENCES

Adar, E., Karger, D. & Stein, A.L. (1999). Haystack: Per-user information environments. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CKIM '99)*. New York: ACM Press. pp. 413–422.

Agarwala, A. & Balakrishnan, R. (2006). Keepin' it real: Pushing the desktop metaphor with physics, piles, and the pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computer Systems (CHI '06)*. New York: ACM Press. pp. 1283–1292.

Apple Computer, Inc. (1995). *OpenDoc Programmer's Guide for the Mac OS.* Reading, Massachusetts: Addison-Wesley.

Bannon, L., Cypher, A., Greenspan, S. & Monty, M.L. (1983). Evaluation and analysis of users' activity organization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '83)*. New York: ACM Press. pp. 54–57.

Bardram, J.E. (1997). Plans as situated action: An activity theory approach to workflow systems. In *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW '97)*. Dordrecht, The Netherlands: Kluwer Academic Publishers. pp. 17–32.

Bardram, J.E. (2005, September). Activity-based computing: Support for mobility and collaboration in ubiquitous computing. *Personal and Ubiquitous Computing, 9*(5), 312–322.

Bardram, J.E. (2007). From desktop task management to ubiquitous activity-based computing. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 223–259). Cambridge, Massachusetts: MIT Press.

Barreau, D. & Nardi, B.A. (1995). Finding and reminding: File organization from the desktop. *ACM SIGCHI Bulletin, 27*, 39–43.

Beaudouin-Lafon, M. & Lassen, H.M. (2000). The architecture and implementation of CPN2000, a post-WIMP graphical application. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '00*). New York: ACM Press. pp. 181–190.

Bellotti, V., Ducheneaut, N., Howard, M. & Smith, I. (2003). Taking email to task: The design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. New York: ACM Press. pp. 345–352.

Bergman, O., Beyth-Marom, R. & Nachmias, R. (2006). The project fragmentation problem in personal information management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. New York: ACM Press. pp. 271–274.

Boer, N., van Baalen, P.J. & Kumar, K. (2002). An activity theory approach for studying the situatedness of knowledge sharing. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35)*. Los Alamitos, California: IEEE Computer Society Press. pp. 1483–1492.

Büscher, M., Mogensen, P., Shapiro, D. & Wagner, I. (1999). The Manufaktur: Supporting work practice in (landscape) architecture. In *Proceedings of the Sixth European Conference on Computer Supported Cooperative Work (ECSCW '99)*. Dordrecht, The Netherlands: Kluwer Academic Publishers. pp. 21–40.

Card, S.K., Pavel, M. & Farrell, J.E. (1984). Window-based computer dialogues. In *Proceedings of the 1st IFIP International Conference on Human-Computer Interaction (INTERACT '84)*. Amsterdam: North-Holland. pp. 239–243.

Cohen, P., Cheyer, A., Wang, M. & Baeg, S.C. (1994). An open agent architecture. In M.N. Huhns & M.P. Singh (Eds.), *Readings in Agents*. San Francisco: Morgan Kaufmann. pp. 197–204.

Czerwinski, M., Cutrell, E. & Horvitz, E. (2000). Instant messaging: Effects of relevance and time. In *People and Computers XIV: Proceedings of HCI 2000* (Vol. 2). Swindon, UK: British Computer Society. pp. 71–76.

Czerwinski, M., Horvitz, E. & Wilhite, S. (2004). A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. New York: ACM Press. pp. 175–182.

Dey, A.K., Salber, D. & Abowd, G.D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCI) Journal, 16*(2–4), 97–166.

Dourish, P., Edwards, W.K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B. & Thornton, J. (2000, April). Extending document management systems with user-specific active properties. *ACM Transactions on Information Systems, 18*(2), 140–170.

Dragunov, A.N., Dietterich, T.G., Johnsrude, K., McLaughlin, M., Li, L. & Herlocker, J.L. (2005). TaskTracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. New York: ACM Press. pp. 75–82.

Drucker, P.F. (1974). *Management: Tasks, Responsibilities, Practices* (1st ed.). New York: Harper & Row.

Ducheneaut, N. & Bellotti, V. (2001, September–October). E-mail as habitat: An exploration of embedded personal information management. *Interactions, 8*(5), 30–38.

Engeström, Y. (1987). *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Helsinki: Orienta-Konsultit Oy.

Fisher, D. & Nardi, B. (2007). Soylent and ContactMap: Tools for constructing the social workscape. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 171–190). Cambridge, Massachusetts: MIT Press.

Fox, A., Johanson, B., Hanrahan, P. & Winograd, T. (2000, May). Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications, 20*(3), 54–65.

Freeman, E. & Fertig, S. (1995). Lifestreams: Organizing your electronic life. In R. Burke (Ed.), *AI Applications in Knowledge Navigation and Retrieval: Papers from the AAAI Fall Symposium (Technical Report FS-95-03)*. Menlo Park, California: AAAI Press.

Freeman, E. & Gelernter, D. (2007). Beyond Lifestreams: The inevitable demise of the desktop metaphor. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 19–48). Cambridge, Massachusetts: MIT Press.

Furnas, G.W. (1986). Generalized fisheye views. In M. Mantei & P. Orbeton (Eds.), *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*. New York: ACM Press. pp. 16–23.

Gay, G. & Hembrooke, H. (2004). *Activity-Centered Design: An Ecological Approach to Designing Smart Tools and Usable Systems*. Cambridge, Massachusetts: MIT Press.

Geyer, W., Vogel, J., Cheng, L. & Muller, M. (2003). Supporting activity-centric collaboration through peer-to-peer shared objects. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '03)*. New York: ACM Press. pp. 115–124.

González, V.M. & Mark, G. (2004). "Constant, constant, multi-tasking craziness": Managing multiple working spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. New York: ACM Press. pp. 113–120.

Gwizdka, J. (2002). TaskView: Design and evaluation of a task-based email interface. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 2002)*. Toronto: International Business Machines, Inc. pp. 4-13.

Halverson, C.A. (2001). Activity theory and distributed cognition: Or what does CSCW need to do with theories? *Computer Supported Cooperative Work, 11*(1–2), 243–267.

Harrison, B.L., Cozzi, A. & Moran, T.P. (2005). Roles and relationships for unified activity management. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '05)*. New York: ACM Press. pp. 236–245.

Haythornthwaite, C., Wellman, B. & Mantei, M. (1995). Work relationships and media use: A social network analysis. *Group Decision and Negotiation, 4*, 193–211.

Henderson, D.A. & Card, S.K. (1986, July). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics, 5*(3), 211–243.

Hong, J.I. & Landay, J.A. (2000). SATIN: A toolkit for informal ink-based applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '00)*. New York: ACM Press. pp. 63–72.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D. & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)*. San Francisco: Morgan Kaufmann. pp. 256–265.

Hutchings, D.R., Czerwinski, M., Robbins, D., Robertson, G., Smith, G. & Meyers, B. (2005). TaskZones: A task manager for multiple-monitor systems. Poster presented at the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05), Seattle, Washington, October 23–26.

Hutchings, D.R. & Stasko, J. (2004). Revisiting display space management: Understanding current practice to inform next-generation design. In *Proceedings of the 2004 Conference on Graphics Interface (GI 2004)*. Waterloo, Ontario: Canadian Human-Computer Communications Society. pp. 127–134.

Hutchins, E. (1995). *Cognition in the Wild.* Cambridge, Massachusetts: MIT Press.

Igarashi, T., Edwards, W.K., LaMarca, A. & Mynatt, E.D. (2000). An architecture for pen-based interaction on electronic whiteboards. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*. New York: ACM Press. pp. 68–75.

Iqbal, S.T., Adamczyk, P.D., Zheng, X.S. & Bailey, B.P. (2005). Towards an index of opportunity: Understanding changes in mental workload during task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. New York: ACM Press. pp. 311–320.

Ishii, H. & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. New York: ACM Press. pp. 234–241.

Kandogan, E. & Schneiderman, B. (1997). Elastic windows: Evaluation of multi-window operations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. New York: ACM Press. pp. 250–257.

Kaptelinin, V. (2003). UMEA: Translating interaction histories into project contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. New York: ACM Press. pp. 353–360.

Kaptelinin, V. & Boardman, R. (2007). Toward integrated work environments: Application-centric versus workspace-level design. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 295–331). Cambridge, Massachusetts: MIT Press.

Kaptelinin, V. & Nardi, B.A. (2006). *Acting with Technology: Activity Theory and Interaction Design.* Cambridge, Massachusetts: MIT Press.

Kaptelinin, V., Nardi, B.A. & Macaulay, C. (1999, July–August). The activity checklist: A tool for representing the "space" of context. *Interactions, 6*(4), 27–39.

Karger, D.R. (2007). Haystack: Per-user information environments based on semistructured data. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 49–100). Cambridge, Massachusetts: MIT Press.

Kidd, A. (1994). The marks are on the knowledge worker. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. New York: ACM Press. pp. 186–191.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B. & Spasojevic, M. (2002). People, places, things: Web presence for the real world. *Mobile Networks and Applications, 7*, 365–376.

Kirsh, D. (2001). The context of work. *Human-Computer Interaction, 16*, 305–322.

Lansdale, M. (1998). The psychology of personal information management. *Applied Ergonomics, 19*(1), 55–66.

Leont'ev, A.N. (1978). *Activity, Consciousness, and Personality* (M.J. Hall, Trans.). Englewood Cliffs, New Jersey: Prentice-Hall.

Light, J. & Miller, J.D. (2002). Miramar: A 3D workplace. In *Proceedings of the IEEE International Professional Communication Conference (IPCC 2002)*. Piscataway, New Jersey: IEEE Press. pp. 271–282.

MacIntyre, B. & Feiner, S. (1996). Language-level support for exploratory programming of distributed virtual environments. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology (UIST '96)*. New York: ACM Press. pp. 83–94.

MacIntyre, B., Mynatt, E.D., Voida, S., Hansen, K.M., Tullio, J. & Corso, G.M. (2001). Support for multitasking and background awareness using interactive peripheral displays. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. New York: ACM Press. pp. 41–50.

Malone, T.W. (1983, January). How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems, 1*(1), 99–112.

Mander, R., Salomon, G. & Wong, Y.Y. (1992). A 'pile' metaphor for supporting casual organization of information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. New York: ACM Press. pp. 627–634.

Mark, G., González, V.M. & Harris, J. (2005). No task left behind? Examining the nature of fragmented work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. New York: ACM Press. pp. 320–330.

Miyata, Y. & Norman, D.A. (1986). Psychological issues in support of multiple activities. In D.A. Norman & S.W. Draper (Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction* (pp. 265–284). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Moran, T.P., Chiu, P., Harrison, S., Kurtenbach, G., Minneman, S. & van Melle, W. (1996). Evolutionary engagement in an ongoing collaborative work process: A case study. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW '96)*. New York: ACM Press. pp. 150–159.

Moran, T.P., Cozzi, A. & Farrell, S.P. (2005, December). Unified activity management: Supporting people in e-business. *Communications of the ACM, 48*(12), 67–70.

Morteo, R., González, V.M., Favela, J. & Mark, G. (2004). Sphere Juggler: Fast context retrieval in support of working spheres. In *Proceedings of the 5th Mexican International Conference in Computer Science (ENC '04)*. Los Alamitos, California: IEEE Computer Society Press. pp. 361–367.

Muller, M.J., Geyer, W., Brownholtz, B., Wilcox, E. & Millen, D.R. (2004). One-hundred days in an activity-centric collaboration environment based on shared objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. New York: ACM Press. pp. 375–382.

Mynatt, E.D. (1999). The writing on the wall. In *Proceedings of the 7th IFIP Conference on Human-Computer Interaction (INTERACT '99)*. Amsterdam: IOS Press. pp. 196–204.

Mynatt, E.D., Igarashi, T., Edwards, W.K. & LaMarca, A. (1999). Flatland: New dimensions in office whiteboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. New York: ACM Press. pp. 346–353.

Mynatt, E.D., Igarashi, T., Edwards, W.K. & LaMarca, A. (2000, July–August). Designing an augmented writing surface. *IEEE Computer Graphics and Applications, 20*(4), 55–61.

Nair, R., Voida, S. & Mynatt, E.D. (2005). Frequency-based detection of task switches. In *Proceedings of the 19th Annual Conference of the British HCI Group (HCI 2005)*. Berlin: Springer. pp. 94–99.

Nardi, B.A. (1996). Studying context: A comparison of activity theory, situated action models, and distributed cognition. In B.A. Nardi (Ed.), *Context and Consciousness: Activity Theory and Human-Computer Interaction* (pp. 69–102). Cambridge, Massachusetts: MIT Press.

Nardi, B.A., Whittaker, S. & Schwarz, H. (2002). NetWORKers and their activity in intensional networks. *Computer Supported Cooperative Work, 11*, 205–242.

Nardi, B.A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J. & Hainsworth, J. (2002, April). ContactMap: Integrating communication and information through visualizing personal social networks. *Communications of the ACM, 45*(4), 89–95.

O'Conaill, B. & Frohlich, D. (1995). Timespace in the workplace: Dealing with interruptions. In *Conference Companion on Human Factors in Computing Systems (CHI '95 Extended Abstracts)*. New York: ACM Press. pp. 262–263.

Olson, J.S., Grudin, J. & Horvitz, E. (2005). A study of preferences for sharing and privacy. In *Extended Abstracts of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05 Extended Abstracts)*. New York: ACM Press. pp. 1985–1988.

Panko, R.R. (1992). Managerial communication patterns. *Journal of Organizational Computing, 2*(1), 95–122.

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L. & Fuchs, H. (1998). The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. New York: ACM Press. pp. 179–188.

Ravasio, P & Tscherter, V. (2007). User's theories of the desktop metaphor, or why we should seek metaphor-free interfaces. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 265–294). Cambridge, Massachusetts: MIT Press.

Rekimoto, J. (1999). Time-machine computing: A time-centric approach for the information environment. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '99)*. New York: ACM Press. pp. 45–54.

Richardson, T., Stafford-Fraser, Q., Wood, K.R. & Hopper, A. (1998, January–February). Virtual Network Computing. *IEEE Internet Computing, 2*(1), 33–38.

Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. & Gorokhovsky, V. (2000). The Task Gallery: A 3D window manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2000)*. New York: ACM Press. pp. 494–501.

Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D. & Smith, G. (2004). Scalable Fabric: Flexible task management. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '04)*. New York: ACM Press. pp. 85–89.

Robertson, G., Smith, G., Meyers, B., Baudisch, P., Czerwinski, M., Horvitz, E., Robbins, D. & Tan, D. (2007). Explorations in task management on the desktop. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 101–138). Cambridge, Massachusetts: MIT Press.

Roseman, M. & Greenberg, S. (1996). TeamRooms: Network places for collaboration. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW '96)*. New York: ACM Press. pp. 325–333.

Scott, S.D., Carpendale, M.S.T. & Inkpen, K.M. (2004). Territoriality in collaborative tabletop workspaces. In *Proceedings of the ACM Conference on Computer Supporter Cooperative Work (CSCW '04)*. New York: ACM Press. pp. 294–303.

Shen, H. & Dewan, P. (1992). Access control for collaborative environments. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work (CSCW '92)*. New York: ACM Press. pp. 51–58.

Sikkel, K. (1997). A group-based authorization model for cooperative systems. In *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW '97)*. Dordrecht, The Netherlands: Kluwer Academic Publishers. pp. 345–360.

Singh, A. (2008, January 25). An introduction to virtualization. [On-line]. Available: http://www.kernelthread.com/publications/virtualization/

Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D. & Andrews, D. (2003). GroupBar: The TaskBar evolved. In *Proceedings of OZCHI 2003*. Brisbane, Australia: University of Queensland. pp. 34–43.

Sousa J.P. & Garlan, D. (2002). Aura: An architectural framework for user mobility in ubiquitous computing environments. In *Software Architecture: System Design, Development and Maintenance, 3rd IEEE/IFIP Conference on Software Architecture (WICSA3)*. Dordrecht, The Netherlands: Kluwer Academic Publishers. pp. 29–43.

Sproull, L.S. (1984). The nature of managerial attention. *Advances in Information Processing in Organizations, 1*, 9–27.

Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P. & Steinmetz, R. (1999). i-LAND: An interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. New York: ACM Press. pp. 120–127.

Stumpf, S., Bao, X., Dragunov, A., Dietterich, T.G., Herlocker, J., Johnsrude, K., Li, L. & Shen, J. (2005). *Predicting user tasks: I know what you're doing!* Paper presented at the Workshop on Human Comprehensible Machine Learning, 20th National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, Pennsylvania, June 9.

Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge, UK: Cambridge University Press.

Szirmay-Kalos, L. (1994). Dynamic layout algorithm to display general graphs. In P.S. Heckbert (Ed.), *Graphics Gems IV* (pp. 505–517). San Diego, California: Academic Press.

Tan, D.S., Meyers, B. & Czerwinski, M. (2004). WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. New York: ACM Press. pp. 1525–1528.

Tang, J.C., Drews, C., Smith, M., Wu, F., Sue, A. & Lau, T. (2007). Exploring patterns of social commonality among file directories at work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. New York: ACM Press. pp. 951–960.

Tashman, C. (2006). WindowScape: A task-oriented window manager. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '06)*. New York: ACM Press. pp. 77–80.

Voida, S., Edwards, W.K., Newman, M.W., Grinter, R.E. & Ducheneaut, N. (2006). Share and share alike: Exploring the user interface affordances of file sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. New York: ACM Press. pp. 221–230.

Voida, S., Mynatt, E.D. & MacIntyre, B. (2007). Supporting activity in desktop and ubiquitous computing. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (pp. 195–222). Cambridge, Massachusetts: MIT Press.

Voida, S., Mynatt, E.D., MacIntyre, B. & Corso, G.M. (2002, July–September). Integrating virtual and physical context to support knowledge workers. *IEEE Pervasive Computing, 1*(3), 73–79.

Vygotsky, L.S. & Cole, M. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, Massachusetts: Harvard University Press.

Whittaker, S., Jones, Q. & Terveen, L. (2002). Contact management: Identifying contacts to support long-term communication. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*. New York: ACM Press. pp. 216–225.

Whittaker, S. & Sidner, C. (1996). Email overload: Exploring personal information management of email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '96)*. New York: ACM Press. pp. 276–283.

Wyckoff, P., McLaughry, S.W., Lehman, T.J. & Ford, D.A. (1998, April). TSpaces. *IBM Systems Journal, 37*(3), 454–476.

VITA

STEPHEN VOIDA

VOIDA was born and raised in the southwestern United States. After graduating from high school, he was pleased to learn that he could earn a degree that would legitimize his passion for tinkering with computers and entered the computer science program at Arizona State University in Tempe, Arizona. He received his B.S. degree in August 1999.

Stephen moved to Atlanta to pursue an M.S. degree in Human-Computer Interaction at the Georgia Institute of Technology. At Georgia Tech, he discovered new interests in augmented reality and interaction design for large displays and elected to continue his studies in the Ph.D. program after receiving his Master's degree in 2001. He participated in a variety of industrial internships while working towards his doctoral degree, spending summers at Microsoft, IBM Research, and the Palo Alto Research Center.

Outside the research lab, Stephen enjoys traveling with his wife, Amy, sharpening his skills as an airplane pilot, and watching *The Daily Show* with his cat.